The Dissertation Committee for Rajiv Ashu Khanna
certifies that this is the approved version of the following dissertation:

New Perspectives and Applications for Greedy Algorithms in Machine Learning

Committee:

Joydeep Ghosh, Supervisor

Alexandros G. Dimakis

Oluwasanmi Koyejo

Sanjay Shakkottai

Constantine Caramanis

New Perspectives and Applications for Greedy Algorithms in Machine Learning

by

Rajiv Ashu Khanna

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2018

Dedicated to my parents Jugal and Vandana Khanna, and to my wife Namrta.

Acknowledgments

I wish to thank the multitudes of people for their both deliberate and inadvertent help without which this thesis would not have been a possibility. I am extremely fortunate to have had Prof. Joydeep Ghosh as my advisor. He has been a constant source of guidance in seeking out unchartered areas of potential research directions, and encouragement in the face of inevitable disappointments that consort with original research. He has been very approachable and accessible throughout my time at UT Austin, and his advice has vastly helped me grow as a researcher and a person.

I have been very fortunate to have had interacted with many eminent researchers during the course of my PhD. I would like to thank Prof. Alexandros Dimakis, who I have gain many valuable insights about work and life in the last four years. I do not remember a single conversation with him, howsoever short, that did not lead me into learning something new. His energy and zeal for research are indeed infectious. I am greatly indebted to Prof. Oluwasanmi Koyejo for his patient mentorship and several collaborations that hopefully will extend into the future as well. I thank Prof. Sahand Negahban for sparking my interest in theoretical aspects of machine learning and several valuable discussions we have had on the topic. I am also thankful to Prof. Martin Jaggi and Dr. Anastasios Kyrillidis for teaching me about optimiza-

tion. I am grateful to Been Kim for making me realize the importance of interpretability in machine learning. I am thankful to Prof. Russell Poldrack and Prof Gunnar Ratsch for their valuable inputs.

My PhD life and learning would have been a lot less fruitful were it not for my student collaborators and colleagues. I am thankful to Sreangsu Acharyya, Ayan Acharya, Shalmali Joshi, Murat Kocaoglu, Francesco Locatello, Gideon Dresdner, Erik Lindgren, Karthikeyan Shanmugam, Ajil Jalal, Ethan Elenberg, Joyce Ho, Yubin Park, Michael Motro, Taewen Kim, Ashish Bora, Farzan Memarian, Diego Garcia-Olano, Alan Gee, Avradeep Bhowmik, Suriya Gunasekar, Jette Hendersen, and Rahi Kalantari. I shall cherish the time we spent together for years to come.

New Perspectives and Applications for Greedy Algorithms in Machine Learning

Publication No. _____

Rajiv Ashu Khanna, Ph.D.
The University of Texas at Austin, 2018

Supervisor: Joydeep Ghosh

Approximating probability densities is a core problem in Bayesian statistics, where the inference involves the computation of a posterior distribution. Variational Inference (VI) is a technique to approximate posterior distributions through optimization. It involves specifying a set of *tractable* densities, out of which the final approximation is to be chosen. While VI is traditionally motivated with the goal of tractability, the focus in this dissertation is to use Bayesian approximation to obtain *parsimonious* distributions. With this goal in mind, we develop greedy algorithm variants and study their theoretical properties by establishing novel connections of the resulting optimization problems in parsimonious VI with traditional studies in the discrete optimization literature. Specific realizations lead to efficient solutions for many sparse probabilistic models like Sparse regression, Sparse PCA, Sparse Collective Matrix Factorization (CMF) etc. For cases where existing results are insufficient to provide acceptable approximation guarantees, we extend the optimization

results for some large scale algorithms to a much larger class of functions.The developed methods are applied to both simulated and real world datasets, including high dimensional functional Magnetic Resonance Imaging (fMRI) datasets, and to the real world tasks of interpreting data exploration and model predictions.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data in many scientific and commercial disciplines are increasingly characterized by high dimensions, often more than the number of samples. In such cases, a-priori knowledge gleaned from expertise and experimental evidence are invaluable for recovering meaningful models. In particular, restricting the degrees of freedom through assumptions of sparsity or low rank has become an important design paradigm, enabling the recovery of parsimonious and interpretable results, and improving storage and prediction efficiency for high dimensional problems. In Bayesian models, such restricted degrees of freedom can be captured by incorporating domain constraints into the design of the prior distribution.

The focus of this dissertation is to provide theoretically motivated and well-founded frameworks that enable extraction of interpretable parsimonious structures through the way of selecting a *few* dimensions or examples that *provably* approximate the data in some sense. The underline theme of this thesis is identification of *hard* discrete optimization problems, primarily in Bayesian Variational Inference but more widely applicable in many cases. Thus, this thesis lies at the intersection of optimization and sparse model selection.

Variational Inference is a practical method for approximating a probability density to make inference more tractable. The focus of research in such methods is favored towards development of algorithms, rather than providing requisite theoretical guarantees for those algorithms. This thesis does both - it provides efficient algorithms for wide ranging applications as well as the respective theoretical guarantees by leveraging novel connections to discrete optimization. The main threads of contributions of this thesis are discussed next.

1.1 Submodular Approximate Inference

Selection of the prior distribution remains one of the most perplexing steps of Bayesian model design, and it is often more challenging when incorporating domain constraints. Prior distributions can be designed by combining conditional distributions - each capturing portions of the problem structure, into a hierarchical model. In other cases, researchers design special purpose prior distributions to match the application at hand. In the case of parameter sparsity, an example of the former approach is the *spike and slab* prior [Mitchell and Beauchamp, 1988, Ishwaran and Rao, 2005], and an example of the latter approach is the *horeshoe* prior [Carvalho et al., 2010].

This thesis develops a broad framework for probabilistic modeling when the apriori knowledge are domain constraints, using the principle of maximum entropy. We show that the desired probabilistic model is nothing but the restriction of the unconstrained model to the domain of interest. In cases where

such a restriction is intractable, we propose a family of parameterized approximations indexed by tractable subsets of the domain constraint set. Further, when these subsets are convex, the posterior expectations remain within the domain constraints. This property is especially useful for interpreting results, and is not shared by many other Bayesian models. e.g. expectations from the spike and slab model are not sparse. We focus on the case of structured sparse support constraints which can be enumerated using a matroid or a knapsack. The set of probabilistic models that can be addressed in this way is quite broad, and include standard sparsity, group sparsity, tree sparsity etc. While the optimal inference is intractable in general, we show that approximate inference using selected convex subsets is equivalent to maximizing a submodular function subject to matroidal/knapsack constraints constraints, and propose efficient greedy-like forward selection procedures which provably guarantees within a constant factor factor of the global optimum.

We develop the framework for domain expertise motivated construction of parsimonious priors in Chapter 3. We provide a variational characterization of the restriction of a probability distribution (Section 3.1), and present an algorithmic framework for tractable inference (Section 3.2). This allows one to cast the problem of sparsification by restriction into a discrete optimization problem (Section 3.3) that we show is submodular. Not only does this entail efficient algorithms for a variety of structured sparsity problems, but it also provides respective approximation guarantees for the same. We validate the proposed algorithms vis-a-vis a multitude of established base-

lines (Section 3.4). To illustrate the effectiveness of our approach, we develop a novel variational Expectation Maximization (EM) algorithm that incorporates the developed methodology for sparse factor models in Chapter 4 and present empirical qualitative and quantitative evaluation on a variety of models (Section 4.4) and real-world functional Magnetic Resonance Imaging (fMRI) datasets (Section 4.5).

1.2 Interpreting Black Box Models

As machine learning (ML) methods have become ubiquitous in human decision making, their transparency and interpretability have grown in importance [Varshney, 2016]. Interpretability is particularity important in domains where decisions can have significant consequences. For example, the pneumonia risk prediction case study in Caruana et al. [2015] showed that a more interpretable model could reveal important but surprising patterns in the data that complex models overlooked. The second thread in this thesis is the formulation of discrete optimization for density approximation routines that enable interpretation of black box models.

Studies of human reasoning have shown that the use of examples (prototypes) is fundamental to the development of effective strategies for tactical decision-making [Newell and Simon, 1972, Cohen et al., 1996]. Example-based explanations are widely used in the effort to improve interpretability. A popular research program along these lines is case-based reasoning (CBR) [Aamodt and Plaza, 1994], which has been successfully applied to real-world prob-

lems [Bichindaritz and Marling, 2006]. More recently, the Bayesian framework has been combined with CBR-based approaches in the unsupervised-learning setting, leading to improvements in user interpretability [Kim et al., 2014]. In a supervised learning setting, example-based classifiers have been is shown to achieve comparable performance to non-interpretable methods, while offering a condensed view of a dataset [Bien and Tibshirani, 2011].

In this thesis, we present two methods for prototype/example selection. Both methods employ density approximation algorithms to select a few examples that closely approximate a known data distribution. The first method is geared towards exploratory analysis, so it selects examples to approximate the training data distribution using kernel herding (Section 5.2). The second method uses Bayesian Quadrature and focuses on interpreting model predictions on the test set (Section 5.1). We show that under certain conditions the former problem is submodular while the latter is *weakly* submodular, which is a generalization of submodularity. We further argue that example based learning is incomplete without also selecting model *criticisms* which are data points that the selected examples do not explain well. This is validated by a human experiment. We also present quantitative empirical evaluation of suggested interpretability approaches for various applications (Section 5.4).

1.3 Weak Submodularity

In this thread, we focus on the weak submodular functions, and show that some algorithms specifically designed for submodular functions can be

used for weak submodular functions with generalized guarantees. While the results in this thread are developed with the scope of Variational Inference in mind, they are applicable for a much larger class of functions. As such, to ease exposition and to emphasize the generality of these results, we present these results independently. The results in this section are at the intersection of continuous and discrete optimization.

Greedy algorithms are widely used for problems in machine learning such as feature selection and set function optimization. Unfortunately, for large datasets, the running time of even greedy algorithms can be quite high. This is because for each greedy step we need to refit a model or calculate a function using the previously selected choices and the new candidate.

Two algorithms that are faster approximations to the greedy forward selection were introduced recently [Mirzasoleiman et al., 2013, 2015]. They achieve better performance by exploiting distributed computation and stochastic evaluation respectively. Both algorithms have provable performance guarantees for submodular functions.

We show that divergent from previously held opinion, submodularity is not required to obtain approximation guarantees for these two algorithms (Sections 6.1,6.2). Specifically, we show that a generalized concept of weak submodularity (seen in Chapter 5 for Sequential Bayesian Quadrature) suffices to give multiplicative approximation guarantees. Our result extends the applicability of these algorithms to a larger class of functions. Furthermore, we show that a bounded submodularity ratio can be used to provide data

dependent bounds that can sometimes be tighter also for submodular functions. We empirically validate our work by showing superior performance of fast greedy approximations versus several established baselines on artificial and real datasets.

Furthermore, even for the unlikely case of low rank optimization, where the choice of potential candidate atoms (i.e. rank-1 matrices) for the selected sparse structure is infinite, we show that the discrete optimization problem is weakly submodular, and hence greedy algorithm variants yields similar approximation guarantees. We show that similar to the standard greedy algorithm for support selection [Elenberg et al., 2018], the approximation bounds for large scale algorithms and the low rank greedy optimization are dependent on the *condition number* of the function, specifically the restricted strong convexity and smoothness parameters of the function on a subset of its domain. Further, we also develop a novel relationship between the condition number and subadditivity of the support selection function. For the low rank case, we improve the exisiting bounds for greedy low rank approximation by an exponential factor (Section 6.3). Finally we present empirical evaluation on simulated and real data sets to show the effectiveness of the simple greedy algorithm variants (Section 6.4) .

Chapter 2

Background and Notation

This section discusses some background that is precursory to this thesis. It also includes notation used throughout this thesis and a few basic definitions. Vectors are denoted by lower case $\boldsymbol{x}$ and matrices by capital $\mathbf{X}$. $x_{i,j}$ denotes the $(i,j)^{\text{th}}$ entry of the matrix $\mathbf{X}$. $\boldsymbol{x}_{i,:}$ denotes the i^{th} row of $\mathbf{X}$ and $\boldsymbol{x}_{:,j}$ denotes the j^{th} column. Let $|\mathbf{X}|$ denote the determinant of $\mathbf{X}$. Sets are denoted by sans serif e.g. S. The reals are denoted by $\mathfrak{R}$. $[n]$ denotes the set of integers $\{1,\ldots,N\}$, and $\wp(N)$ denotes the power set of $[n]$.

2.1 Definitions

Definition 2.1.1 (KL divergence)**.** Let X be either a countable set, or a Polish space[1] equipped with the standard Borel σ-algebra of measurable sets, and let $\boldsymbol{Q}$ and $\boldsymbol{P}$ be probability measures on X. The *relative entropy*, also known as the *Kullback-Leibler divergence* (KL divergence) of the probability measure $\boldsymbol{Q}$ with respect to a *base* measure $\boldsymbol{P}$ is given by:

$$\mathrm{KL}(\boldsymbol{Q}\|\boldsymbol{P}) = \int_{\mathsf{X}} \frac{d\boldsymbol{Q}}{d\boldsymbol{P}}(x) \log \frac{d\boldsymbol{Q}}{d\boldsymbol{P}}(x) d\boldsymbol{P}(x),$$

[1] A complete separable metric space.

where $\mathrm{KL}(\boldsymbol{Q}\|\boldsymbol{P})$ is the Radon-Nikodym derivative, existence of which requires that $\boldsymbol{Q}$ is absolutely continuous with respect to $\boldsymbol{P}$.

When $\boldsymbol{P}$ is the uniform measure, the $\mathrm{KL}(\boldsymbol{Q}\|\boldsymbol{P})$ is the negative of the entropy of $\boldsymbol{Q}$. Let $\mathcal{P} \ni p$ denote the set of probability densities on X i.e. positive functions $p : \mathsf{X} \mapsto [0, 1]$ that integrate to one. For the rest of this paper, we assume that $\boldsymbol{P}$ is absolutely continuous with respect to a background measure ν so there exists a density $p \in \mathcal{P}$ that satisfies $d\boldsymbol{P} = pd\nu$. This further implies the existence of a density $q \in \mathcal{P}$ that satisfies $d\boldsymbol{Q} = qd\nu$. To simplify notation, we use use the standard $d\nu = dx$. Thus, the relative entropy is given in terms of the densities as:

$$\mathrm{KL}(q\|p) = \int_{\mathsf{X}} q(x) \log \frac{q(x)}{p(x)} dx.$$

All of the results outlined in this manuscript apply, for example, to Euclidean sample spaces using the Lebesgue measure, and to discrete sample spaces using the counting measure as the background measure ν.

Let $\mathbb{E}$ be the *expectation operator*, which we denote using densities as $\mathrm{E}_p[f] = \int_{\mathsf{X}} p(x)f(x)dx$ to simplify notation. We suppress the dependence on the random variable X when the expectation and the relative entropy are clear from context. The *delta function*, denoted by $\delta : \mathsf{X} \mapsto \Re$, is a generalized set function that satisfies $\int_{\mathsf{X}} \delta_{\mathsf{A}}(x)f(x) = g(x)$, for some some $\mathsf{A} \subseteq \mathsf{X}$ where $g(x) = f(x)\ \forall x \in \mathsf{A}$, and $g(x) = 0$ otherwise. If $\nu(\mathsf{A})$ is finite, δ_{A} is also a density $\delta_{\mathsf{A}} \in \mathcal{P}$, corresponding to a probability measure $\boldsymbol{P}_{\mathsf{A}}$ which assign a probability of 1 to sets B where $\mathsf{A} \cap \mathsf{B} \neq \emptyset$, and is 0 otherwise [Kolmogorov,

1933]. The *domain restricted density set* of $\mathsf{A} \subset \mathsf{X}$, denoted by $\mathcal{F}_{\mathsf{A}}$ is the set of probability density functions supported on A i.e. densities that satisfy $\mathcal{F}_{\mathsf{A}} = \{q \in \mathcal{P} \mid q(x) = 0 \; \forall \; x \notin \mathsf{A}\} \subset \mathcal{P} = \mathcal{F}_{\mathsf{X}}$.

Definition 2.1.2 (Inverse-Indicator)**.** An *inverse-indicator function* of a set $\mathsf{A} \subset \mathsf{X}$, denoted by ϕ_{A} is a function $\phi_{\mathsf{A}} : \mathsf{X} \mapsto \mathfrak{R}_+$ that satisfies $\phi_{\mathsf{A}}(\mathbf{x}) > 0$ for $x \notin \mathsf{A}$, and $\phi_{\mathsf{A}}(\mathbf{x}) = 0$ otherwise.

For example, $\delta_{\mathsf{X}\setminus\mathsf{A}}$ is an inverse-indicator function of A.

Definition 2.1.3 (Information projection)**.** The *information projection* of a probability density p to a constraint set $\mathcal{A}$ is given by the solution of:

$$\inf_{q \in P} \mathrm{KL}(q\|p) \text{ s.t. } q \in \mathcal{A}.$$

We only consider projections where $\mathcal{A} \neq \emptyset$, so the infimum is achieved. We will assume that the base measure $\boldsymbol{P}$ is a probability measure, although all of the presented results are easily extendable to arbitrary positive measures.

Restriction of probability densities[2] is a standard approach for defining distributions on subsets of a sample space. Let p be a probability density function on X. The restriction of the density p to the set $\mathsf{A} \subset \mathsf{X}$ is given by:

$$q(x) = \begin{cases} \frac{p(x)}{\int_{\mathsf{A}} p(x)dx} & x \in \mathsf{A}, \\ 0 & \text{otherwise.} \end{cases}$$

[2] A special case of *restriction of measures* [Kolmogorov, 1933].

An important case is when A is a measure zero subset of X. The conditional density is one such example [Chang and Pollard, 1997], the existence of which follows from the *disintegration theorem* [Kolmogorov, 1933].

For completeness, we also discuss relevant definitions from the discrete optimization literature.

Definition 2.1.4 (Submodular)**.** A set function $F : \wp(d) \mapsto \mathfrak{R}$ is submodular, if for all subsets $\mathsf{u}, \mathsf{v} \subseteq \mathsf{m}$ it holds that $F(\mathsf{u} \cup \mathsf{v}) + F(\mathsf{u} \cap \mathsf{v}) \leq F(\mathsf{u}) + F(\mathsf{v})$.

Definition 2.1.5 (Monotonic)**.** A set function $F : \wp(d) \mapsto \mathfrak{R}$ is monotonic, if for all subsets $\mathsf{u} \subset \mathsf{v} \subseteq \wp(d)$ it holds that $F(\mathsf{u}) \leq F(\mathsf{v})$.

Further, f is *normalized* if $f(\emptyset) = 0$. Submodular functions have a diminishing returns property [Nemhauser et al., 1978] i.e. the marginal gain of adding elements decreases with the size of the set. Submodular functions are of special interest because greedy algorithm and its simple variants achieve provable approximation guarantees for several otherwise NP-Hard combinatorial optimization problems [Nemhauser et al., 1978, Sviridenko, 2004, Calinescu et al., 2011]. An additional benefit of the greedy approach is that it does not require the model selection decision of the sparsity to be made at training time. The greedy algorithm is outlined in Algorithm 1.

Definition 2.1.6. A matroid is a structure (N, E), where N is the *ground set*, and $\mathsf{E} \subset \wp(\mathsf{N})$ is a family of *independent* sets that satisfies: (i) $\mathsf{B} \in \mathsf{E}, \mathsf{A} \subset \mathsf{B} \implies \mathsf{A} \in \mathsf{E}$, and, (ii) $\mathsf{A} \in \mathsf{E}, \mathsf{B} \in \mathsf{E}, |\mathsf{A}| < |\mathsf{B}| \implies \exists x \in \mathsf{B} - \mathsf{A}$ s.t. $\mathsf{A} \cup x \in \mathsf{E}$.

Algorithm 1 Greedy algorithm, $\max F(\mathsf{S})$ s.t. $|\mathsf{S}| \leq k_*$

Input: $k_*, \mathsf{S} = \emptyset$
while $|\mathsf{S}| < k_*$ **do**
  foreach $i \in [n] \backslash \mathsf{S}, f_i = F(\mathsf{S} \cup i) - F(\mathsf{S})$
  $\mathsf{S} = \mathsf{S} \cup \{\arg\max f_i\}$
end while
Return: S.

Matroids are useful in enforcing structured sparsity constraints. A *uniform* matroid has E as the set of all possible k and lesser sized subsets of N, and thus induces the k-cardinality constraint. Similarly, a knapsack constraint can be encoded by a matroid which has each candidate solution in E as a set of possible groups, each with an associated cost, such that the total cost of each candidate solution in E is less than or equal to the knapsack value. A *partition* matroid partitions N into subsets $\{\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_r\}$, with $\mathsf{E} = \{\mathsf{A} \,|\, \mathsf{A} \subset \mathsf{N}, |\mathsf{A} \cap \mathsf{X}_i| \leq k_i \forall i \in [r]\}$ for given $\{k_1, k_2, \ldots, k_r\}$.

Definition 2.1.7 (Submodularity Ratio [Das and Kempe, 2011])**.** Let $\mathsf{S}, \mathsf{L} \subset [p]$ be two disjoint sets, and $f(\cdot) : [p] \to \mathbb{R}$. The submodularity ratio of L with respect to S is given by

$$\gamma_{\mathsf{L},\mathsf{S}} := \frac{\sum_{j \in \mathsf{S}} [f(\mathsf{L} \cup \{j\}) - f(\mathsf{L})]}{f(\mathsf{L} \cup \mathsf{S}) - f(\mathsf{L})}. \tag{2.1}$$

The submodularity ratio of a set U with respect to an integer k is given by

$$\gamma_{\mathsf{U},k} := \min_{\substack{\mathsf{L},\mathsf{S}:\mathsf{L} \cap \mathsf{S} = \emptyset, \\ \mathsf{L} \subseteq \mathsf{U}, |\mathsf{S}| \leq k}} \gamma_{\mathsf{L},\mathsf{S}}. \tag{2.2}$$

It is easy to show that $f(\cdot)$ is submodular if and only if $\gamma_{\mathsf{L},\mathsf{S}} \geq 1$ for all sets L and S. However, an approximation guarantee is obtained when

$0 < \gamma_{\mathsf{L},\mathsf{S}} \ \ \forall\, \mathsf{L}, \mathsf{S}$ [Das and Kempe, 2011, Elenberg et al., 2018]. The subset of monotone functions which have $\gamma_{\mathsf{L},\mathsf{S}} > 0 \ \ \forall\, \mathsf{L}, \mathsf{S}$ are called weakly submodular functions in the sense that even though the function is not submodular, it still provides a provable bound for greedy selections.

Also vital to our analysis is the notion of restricted strong concavity and smoothness [Negahban et al., 2012, Loh and Wainwright, 2015].

Definition 2.1.8 (Low Rank Restricted Strong Concavity (RSC), Restricted Smoothness (RSM))**.** A function $\ell : \mathbb{R}^{n\times d} \to \mathbb{R}$ is said to be restricted strong concave with parameter m_Ω and restricted smooth with parameter M_Ω if for all $\mathbf{X}, \mathbf{Y} \in \Omega \subset \mathbb{R}^{n\times d}$,

$$\begin{aligned} -\frac{m_\Omega}{2}\|\mathbf{Y}-\mathbf{X}\|_F^2 &\geq \ell(\mathbf{Y}) - \ell(\mathbf{X}) - \langle \nabla\ell(\mathbf{X}), \mathbf{Y}-\mathbf{X}\rangle \\ &\geq -\frac{M_\Omega}{2}\|\mathbf{Y}-\mathbf{X}\|_F^2. \end{aligned}$$

If a function $\ell(\cdot)$ has restricted strong concavity parameter m, then its negative $-\ell(\cdot)$ has restricted strong convexity parameter m. We choose to use the nomenclature of concavity for ease of exposition in terms of relationship to submodular maximization. Further, note that we define RSC/RSM conditions on the space of matrices rather than vectors, on a domain Ω constrained by rank rather than sparsity. It is straightforward to see that if $\Omega' \subseteq \Omega$, then $M_{\Omega'} \leq M_\Omega$ and $m_{\Omega'} \geq m_\Omega$.

2.2 Maximum Mean Discrepancy (MMD)

The maximum mean discrepancy (MMD) is a measure of the difference between distributions P and Q, given by the suprenum over a function space $\mathcal{F}$ of differences between the expectations with respect to two distributions. The MMD is given by:

$$\mathrm{MMD}(\mathcal{F}, P, Q) = \sup_{f \in \mathcal{F}} \left(\mathrm{E}_{X \sim P} \left[f(X) \right] - \mathrm{E}_{Y \sim Q} \left[f(Y) \right] \right). \tag{2.3}$$

When $\mathcal{F}$ is a reproducing kernel Hilbert space (RKHS) with kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, the suprenum is achieved at [Gretton et al., 2012]:

$$f(x) = \mathrm{E}_{X' \sim P} \left[k(x, X') \right] - \mathrm{E}_{X' \sim Q} \left[k(x, X') \right]. \tag{2.4}$$

The function (2.4) is also known as the *witness function* as it measures the maximum discrepancy between the two expectations in $\mathcal{F}$. Observe that the witness function is positive whenever Q underfits the density of P, and negative wherever Q overfits P. We can substitute (2.4) into (2.3) and square the result, leading to:

$$\mathrm{MMD}^2(\mathcal{F}, P, Q) = \mathrm{E}_{X, X' \sim P} \left[k(X, X') \right] - 2\mathrm{E}_{X \sim P, y \sim Q} \left[k(X, Y) \right] + \mathrm{E}_{Y, Y' \sim Q} \left[k(Y, Y') \right]. \tag{2.5}$$

It is clear that $\mathrm{MMD}^2(\mathcal{F}, P, Q) \geq 0$ and $\mathrm{MMD}^2(\mathcal{F}, P, Q) = 0$ iff. P is indistinguishable from Q on the RHKS $\mathcal{F}$. This population definition can be approximated using sample expectations. In particular, given n samples from P as $\mathsf{X} = \{x_i \sim P,\, i \in [n]\}$, and m samples from Q as $\mathsf{Z} = \{z_i \sim Q,\, i \in [m]\}$,

the following is a finite sample approximation:

$$\mathrm{MMD}_b^2(\mathcal{F}, \mathsf{X}, \mathsf{Z}) = \frac{1}{n^2} \sum_{i,j \in [n]} k(x_i, x_j) - \frac{2}{nm} \sum_{i \in [n], j \in [m]} k(x_i, z_j) + \frac{1}{m^2} \sum_{i,j \in [m]} k(z_i, z_j), \tag{2.6}$$

and the witness function is approximated as:

$$f(x) = \frac{1}{n} \sum_{i \in [n]} k(x, x_i) - \frac{1}{m} \sum_{j \in [m]} k(x, z_j). \tag{2.7}$$

Chapter 3

Constructing Parsimonious Priors

In this chapter, we focus on the construction of a broad framework for probabilistic modeling under domain constraints. Our framework follows the *maximum entropy principle* [Jaynes, 1957]. According to this principle, the probability distribution is selected as one that satisfies the constraints while remaining as *random* as possible. Thus the selected distribution incorporates known information, but makes no further assumptions. When the sample space includes a base measure, the principle is known as the *principle of minimum discrimination information* [Kullback, 1959]. Here, the distribution is chosen as one that satisfies the constraints, but is as difficult as possible to discriminate from the base measure when the difference is measured by relative entropy.

The maximum entropy approach has been especially successful when knowledge can be expressed as expectation constraints. In such cases, it has been shown that the solution is given by a member of the exponential family [MacKay, 2003, Koyejo, 2013] e.g. quadratic constraints result in the Gaussian distribution, and linear expectation constraints (for positive variables) result in the exponential distribution. Similarly, other members of the exponential

family such as the Bernoulli, Poisson and Gamma distributions may be motivated as solutions of appropriate expectation constrained relative entropy minimization problems. The purpose of our work is to extend this approach to the design of probabilistic models when the knowledge is encoded as a wider class of domain constraints.

Our main theoretical and algorithmic contributions are:

- We prove that the information projection of a density to domain constraints is given by its restriction (Section 3.1).
- We characterize the restriction precisely, showing that it is given by a conditional distribution (Section 3.3).
- We propose a family of parameterized approximations indexed by tractable subsets of the domain constraints when the optimal inference is intractable (Section 3.2). In the special case of sparsity constraints that admit an enumeration using a matroid or knapsack, we consider approximate inference using convex sparse support sets.
- We show that the sparse support estimation problem is submodular. As a result, greedy forward selection is efficient and guarantees constant factor optimality (Section 3.3.1).
- To illustrate the derivation of an efficient algorithm, we employ the Gaussian base measure induced by knowledge of local partial correlation and

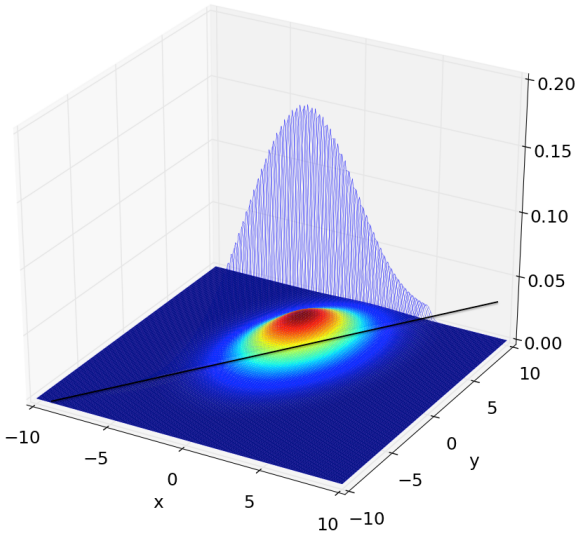


Figure 3.1: Multivariate Gaussian density and its restriction to the diagonal line shown.

consider the design of priors that capture the domain knowledge of sparse support (Section 3.3.3).

3.1 Probabilistic Inference with Domain Constrained Variables

Let X be the random variable of interest, and p represent the base measure. We assume a-priori information identifying a sub-domain $\mathsf{A} \subset \mathsf{X}$. Following the *principle of minimum discrimination information*, the prior is chosen as the *information projection* of the base density p to $\mathcal{F}_{\mathsf{A}}$. To begin, we show that the support constraint is equivalent to a particular expectation constraint.

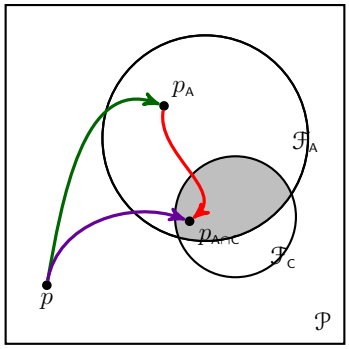


Figure 3.2: Equivalence between the sequence of information projections $\mathcal{P}o\mathcal{F}_{\mathsf{A}}o\mathcal{F}_{\mathsf{C}}$ and $\mathcal{P}o\mathcal{F}_{\mathsf{A}\cap\mathsf{C}}$ Note that this property need not hold for non-domain constraints. (Theorem 3.2.1).

Lemma 3.1.1. *Let $\mathcal{F}_{\mathsf{A}}$ be the domain restricted density set of A, ϕ_{A} be its inverse-indicator function, and $\mathcal{G} = \{q \in \mathcal{P} \mid \mathrm{E}_q[\phi_{\mathsf{A}}(x)] = 0\} \subset \mathcal{P}$. Then $\mathcal{F}_{\mathsf{A}} = \mathcal{G}$.*

Proof. $[\mathcal{F}_{\mathsf{A}} \subset \mathcal{G}]$: Let $q \in \mathcal{F}_{\mathsf{A}}$, then $\mathrm{E}_q[\phi_{\mathsf{A}}] = 0$, thus $q \in \mathcal{G}$. $[\mathcal{G} \subset \mathcal{F}_{\mathsf{A}}]$: Let $q \in \mathcal{G}$, the non-negativity of ϕ_{A} implies that for each $x \in \mathsf{X}$, either $q = 0$ or $\phi_{\mathsf{A}} = 0$. □

The following Lemma characterizes relative entropy minimization subject to norm ball expectation constraints. For simplicity, the theorem is modified to address the special case of the result where a solution exists and the infimum is attained.

Lemma 3.1.2 ([Altun and Smola, 2006])**.**

$$\begin{aligned} & \min_{q \in \mathcal{P}} \mathrm{KL}(q\|p) \; s.t. \; \mathrm{E}_q\left[\boldsymbol{\beta}\right] = \mathbf{b} \\ = \; & \max_{\boldsymbol{\lambda}} \langle \boldsymbol{\lambda}, \mathbf{b} \rangle - \log \int_{\mathsf{X}} p(x) e^{\langle \boldsymbol{\lambda}, \boldsymbol{\beta}(x) \rangle} dx + e^{-1}, \end{aligned}$$

and the unique solution is given by $q_*(x) = p(x)e^{\langle \boldsymbol{\lambda}_*, \boldsymbol{\beta}(z) \rangle - G(\boldsymbol{\lambda}_*)}$ *where* $\boldsymbol{\lambda}_*$ *is the dual solution and* $G(\boldsymbol{\lambda}_*)$ *ensures normalization.*

We can now show our first main result, investigating the relationship between restriction of densities and information projection subject to domain constraints.

Theorem 3.1.3. *The information projection of the base density* p *to the constraint set* $\mathcal{F}_{\mathsf{A}}$ *is the restriction of* p *to the domain* A*.*

Proof. The information projection of p to $\mathcal{F}_{\mathsf{A}}$ is equivalent to (Lemma 3.1.1):

$$\min_{q \in \mathcal{P}} \mathrm{KL}(q\|p) \text{ s.t. } \mathrm{E}_q\left[\phi_{\mathsf{A}}\right] = 0. \tag{3.1}$$

Following Lemma 3.1.2, the solution is given by $q_*(z) = p(x)e^{\langle \lambda_*, \phi_{\mathsf{A}} \rangle - G(\lambda_*)}$, where:

$$\lambda_* = \arg\max_{\lambda} \langle \lambda, 0 \rangle - \log \int_{\mathbb{X}} p(x) e^{\langle \lambda, \phi_{\mathsf{A}}(x) \rangle} dx.$$

Clearly $\lambda_* = -\infty$, thus $e^{\langle \lambda_*, \phi_{\mathsf{A}}(x) \rangle} \to \delta_{\mathsf{A}}(x)$ via standard limit arguments, so $q_* = p(x)\delta_{\mathsf{A}}(x) / \int_{\mathsf{A}} p(x)dx$. □

In other words, Theorem 3.1.3 shows the equivalence between information projection subject to domain constraints and density restriction. The-

orem 3.1.3 also generalizes known results. Given the prior $p(x)$ and likelihood $p(y|x)$, let $\hat{y} \in \mathsf{Y}$ represent the observation(s). Williams [1980] showed that the Bayesian posterior can be recovered as the information projection of $p(x, y) = p(y|x)p(x)$ to $\mathcal{F}_{\mathsf{X}\times\{\hat{y}\}}$, and the solution is given by $p(x|y = \hat{y})\delta_{\hat{y}}$. Our result gives an alternative proof of this result as a special case of Theorem 3.1.3, which we present next.

Corollary 3.1.4. *Consider the product space* $\mathsf{X} = \mathsf{W} \times \mathsf{Y}$. *Let domain constraint be given by* $\mathsf{W} \times \{\hat{y}\}$ *for some* $\hat{y} \in \mathsf{Y}$. *The information projection of* p *to* $\mathcal{F}_{\mathsf{W}\times\{\hat{y}\}}$ *is given by* $p(w|y = \hat{y})\delta_{\hat{y}}$.

Theorem 3.1.3 gives principled motivation for the domain restriction approach to structured prior design. This approach has proven to be quite useful in various contexts. Various properties of the restriction, such as its shape, and tail behavior (up to rescaling) follow directly from the base density, thus if it is often easy to analyze the properties of the prior when the base measure is well understood. Examples of density restriction in the literature include the truncated Gaussian, Beta and Gamma densities [Damien and Walker, 2001], and the restriction of the matrix variate Gaussian to the manifold of low rank matrices [Park and Pillow, 2013].

3.2 Approximate Inference via Tractable Subsets

For many domains of interest, the normalization constant required for domain restriction is computationally intractable. In theory, rejection sam-

pling methods and Markov Chain Monte Carlo (MCMC) inference methods [Robert et al., 1999] do not require knowledge of this value. However, it is often the case that the constrained domains are measure zero sets with respect to the base measure. In practice, this means that random samples generated from the base measure are unlikely to lie in the constrained domains e.g. random samples from a matrix Gaussian will rarely be low rank. In such cases, rejection sampling fails, and MCMC suffers from low acceptance probabilities. Thus inference with such domain constraints typically requires specialized methods e.g. [Damien and Walker, 2001, Park and Pillow, 2013]. In the following, we propose a class of variational approximations, based on an inner representation of the domain constraint set.

Let $\{\mathsf{S}_i \in \mathsf{A}\}$ represent a (possibly overlapping) partitioning of A into subsets. We define the density support sets generated by these partitions as $\mathcal{F}_{\mathsf{S}_i}$, and their union $\mathcal{D} = \bigcup \mathcal{F}_{\mathsf{S}_i}$. Note that by definition each $\mathcal{F}_{\mathsf{S}_i} \subseteq \mathcal{D} \subseteq \mathcal{F}_{\mathsf{A}} \subseteq \mathcal{F}_{\mathsf{X}}$. We will use $\mathcal{D}$ as the approximation set. Our approach results in a parameterized set of prior densities p_{S} corresponding to choices of S, combined with an information projection rule for selecting the p_{S} from the domain restricted density set $\mathcal{F}_{\mathsf{S}}$.

While our approach allows for the choice of S_* to be based on the prior alone, such a result will most accurately capture capture the prior mass, instead of the posterior mass. Thus we consider approximations that incorporate the data likelihood via the posterior. Let $p(y|x)$ represent the observed data likelihood, and $p(x|y)$ represent the posterior. $p(y)$ is known as the evidence.

We select the approximate prior as the solution of:

$$\min_{q \in \mathcal{D}} \mathrm{KL}\left(\frac{p(y|x)q(x)}{Z} \middle\| \frac{p(y|x)p_{\mathsf{A}}(x)}{p_{\mathsf{A}}(y)} \right), \tag{3.2}$$

where the normalization factors are given by $p_{\mathsf{A}}(y) = \int_{\mathsf{X}} p(y|x)p_{\mathsf{A}}(x)dx$ and $Z = \int_{\mathsf{X}} p(y|x)q(x)dx$. The approximate prior is then given by $p_{\mathsf{S}_*}(x)$, based on the corresponding partition S_*, and the posterior approximation is given by $p_{\mathsf{S}_*}(y|x) \propto p(y|x)p_{\mathsf{S}_*}(x)$. Thus, the proposed approximation is most useful when there exist subsets of A such that the restriction of the base density to each subset leads to tractable inference, the result is most accurate when one of the subsets $\mathsf{S}_* \in \mathsf{A}$ captures most of the posterior probability mass.

In the following, we will show that (3.2) can be simplified and inference does not require knowledge of the intractable prior density. First we consider a general result for a sequence of information projections, the result is also illustrated in Fig. 3.2.

Theorem 3.2.1. *Let $\pi : [n] \mapsto [n]$ be a permutation function and $\{\mathsf{C}_{\pi(i)} \mid \mathsf{C}_{\pi(i)} \subset \mathsf{X}\}$ represent a sequence of sets with non empty intersection $\mathsf{B} = \bigcap \mathsf{C}_i \neq \emptyset$. Given a base density p, let $q_0 = p$, and define the sequence of information projections:*

$$q_i = \operatorname*{arg\,min}_{q \in \mathcal{F}_{\mathsf{C}_{\pi(i)}}} \mathrm{KL}(q \| q_{i-1}),$$

then $q_* = q_N$ is independent of π. Further:

$$q_* = \min_{q \in \mathcal{F}_{\mathsf{B}}} \mathrm{KL}(q \| p).$$

Proof. Consider the case when $N = 2$, then $\pi : [1, 2] \mapsto \{[1, 2], [2, 1]\}$. From Theorem 3.1.3, we have that $q_2(x) \propto p(x)\delta_{\mathsf{C}_1}(x)\delta_{\mathsf{C}_2}(x) = p(x)\delta_{\mathsf{B}}(x)$ independent of π. The proof is extended to $N > 2$ by induction. □

We emphasize that this proof depends on the equivalence between information projection unto support constraints and restriction of distributions, this the result does not necessarily hold for more general information projections.

We apply Theorem 3.1.3 to formulate equivalent solutions for approximate inference that may be simpler to solve.

Corollary 3.2.2. *Let $p_{\mathsf{S}_*}(x)$ represent the prior density selected as the minimizer of the minimizer of* (3.2)*, then:*

$$p_{\mathsf{S}_*}(x) = \min_{q(x)\in\mathcal{D}} \mathrm{KL}\left(\frac{p(y|x)q(x)}{Z}\middle\|\frac{p(y|x)p(x)}{p(y)}\right), \tag{3.3}$$

where $Z = \int_{\mathsf{X}} p(y|x)q(x)dx$. Further, let $p_{\mathsf{S}_*}(x|y)$ be the corresponding posterior density, then:

$$p_{\mathsf{S}_*}(x|y) = \arg\min_{q\in\mathcal{D}} \mathrm{KL}(q(x)\|p_{\mathsf{A}}(x|y)) \tag{3.4}$$

$$= \arg\min_{q\in\mathcal{D}} \mathrm{KL}(q(x)\|p(x|y)). \tag{3.5}$$

Proof. In this proof, we consider the joint sample space $\mathsf{X} \times \mathsf{Y}$. The optimal posterior $p(y|x)p_{\mathsf{S}_*}(x)$ from (3.2) is the information projection of the optimal prior $p(x)p_{\mathsf{S}_*}(x)$ to $\mathcal{F}_{\mathsf{X}\times\{\hat{y}\}}$. It follows, from the strict convexity of relative entropy, that this mapping is onto, thus (3.2) is equivalent to direct optimization

of the posterior over the set $\mathcal{F}_{\mathsf{X}\times\{\hat{y}\}} \cap \mathcal{F}_{\mathsf{D}\times\{\hat{y}\}}$. This corresponds to a sequence of domain projections, thus by Theorem 3.1.3, the resulting posterior is equivalent to the solution of (3.4). The equivalence between the solutions of (3.3) and (3.5) can be shown using identical steps. (3.5) corresponds to the projection of $p(x)p(y|x)$ to the set $\mathcal{F}_{\mathsf{X}} \times \mathcal{F}_{\{\hat{y}\}}$ followed by projection to $\mathcal{D} \times \mathcal{F}_{\{\hat{y}\}}$, and $\mathcal{F}_{\mathsf{A}} \cap \mathcal{D} = \mathcal{D}$, so the solutions are equivalent by Theorem 3.1.3. □

3.3 Priors for Sparse Variables

For ease of exposition, we begin with a special case of the proposed framework where the sample space $\mathsf{X} = \tilde{\mathsf{X}}^d$ - a d dimensional product space and domain constraints correspond to a-priori knowledge about sparsity. A d dimensional variable $\boldsymbol{x} \in \mathsf{X}$ is *k-sparse* if at least k of its entries take a *default* value of $c_i \in \tilde{\mathsf{X}}_i$ i.e $|\{i \mid x_i = c_i\}| \geq k$. In Euclidean space $\mathsf{X} = \mathfrak{R}^d$ and in most cases, $c_i = c = 0 \ \forall \ i$. The *support* of $\boldsymbol{x} \in \mathsf{X}$ is the set $\text{supp}\,(\boldsymbol{x}) = \{i \mid x_i \neq c_i\} = \mathsf{s} \in \wp(d)$.

Let $\mathsf{S} \subset \mathsf{X}$ denote the set of variables with support s i.e. $\mathsf{S} = \{\boldsymbol{x} \in \mathsf{X} \text{ s.t. } \text{supp}\,(\boldsymbol{x}) = \mathsf{s}\}$. We will use the notation $\boldsymbol{x}_{\mathsf{s}} = \{x_i \mid i \in \mathsf{s}\}$, and its complement $\boldsymbol{x}_{\mathsf{s}'} = \{x_i \mid i \in \mathsf{s}'\}$, where $\mathsf{s}' = [d] \backslash \mathsf{s}$. The domain of k sparse vectors is given by the union of all possible $\frac{d!}{(d-k)!k!}$ sparse support sets as $\mathsf{A} = \bigcup \mathsf{S}_i$. While the sparse domain A is non-convex, each subset S is a convex set, in fact given by linear subspaces with basis $\{e_i \mid i \in \mathsf{s}\}$. Further, while the information projection of a base density p to A is generally intractable, the information projection to its convex subsets S turn out to be computationally

tractable. We will investigate the proposed approximation scheme using these subsets.

Consider the information projection of an arbitrary probability measure $\boldsymbol{P}$ with density[1] p to the set $\mathcal{D} = \bigcup \mathcal{F}_{\mathsf{S}_i}$ given by:

$$\begin{aligned}\min_{q\in\mathcal{D}} \mathrm{KL}(q\|p) &= \min_{\mathsf{S}\in\{\mathsf{S}_i\}} \left[\arg\min_{q\in\mathcal{F}_{\mathsf{S}}} \mathrm{KL}(q\|p)\right] \\ &= \min_{\mathsf{S}\in\{\mathsf{S}_i\}} \mathrm{KL}(p_{\mathsf{S}}\|p),\end{aligned}$$

where p_{S} is the information projection of p to a set $\mathcal{F}_{\mathsf{S}}$. Applying Theorem 3.1.3, we find that $p_{\mathsf{S}} = p(\boldsymbol{x})\delta_{\mathsf{S}}(\boldsymbol{x})/Z$, where Z is a normalization factor:

$$\begin{aligned}Z &= \int_{\mathsf{S}} p(\boldsymbol{x}) = \int_{\mathsf{X}} p(\boldsymbol{x}_{\mathsf{S}}, \boldsymbol{x}_{\mathsf{S}'})\delta_{\mathsf{S}}(\boldsymbol{x}) \\ &= \int_{\mathsf{X}} p(\boldsymbol{x}_{\mathsf{S}}|\boldsymbol{x}_{\mathsf{S}'})p(\boldsymbol{x}_{\mathsf{S}'})\delta_{\mathsf{S}}(\boldsymbol{x}) \\ &= p(\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'}).\end{aligned}$$

Thus, the normalization factor is a marginal density at $\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'}$.

We may now compute the restriction explicitly:

$$\begin{aligned}p_{\mathsf{S}}(\boldsymbol{x}) &= p(\boldsymbol{x}_{\mathsf{S}}|\boldsymbol{x}_{\mathsf{S}'})p(\boldsymbol{x}_{\mathsf{S}'})\delta_{\mathsf{S}}(\boldsymbol{x})/p(\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'}) \\ &= p(\boldsymbol{x}_{\mathsf{S}}|\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'})\delta_{\mathsf{S}}(\boldsymbol{x}). \qquad (3.6)\end{aligned}$$

In other words, the information projection to a sparse support domain is the

[1]Where p may represent the conditional density $p(\boldsymbol{x}|y)$ as in (3.5). To simplify the discussion, we suppress the dependence on y.

conditional distribution of $\boldsymbol{x} \in \mathsf{S}$ at $\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'}$. The resulting gap is:

$$\begin{aligned}
\mathrm{KL}(p_{\mathsf{S}} \| p) &= \int_{\mathsf{S}} p_{\mathsf{S}}(\boldsymbol{x}) \log \frac{p_{\mathsf{S}}(x)}{p(\boldsymbol{x})} \\
&= \int_{\mathsf{S}} p_{\mathsf{S}}(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{p(\boldsymbol{x}) p(\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'})} \\
&= -\log p(\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'}).
\end{aligned}$$

Define $J : \wp(d) \mapsto \mathfrak{R}$, $J(\mathsf{s}) = \log p(\boldsymbol{x}_{\mathsf{S}'} = \boldsymbol{c}_{\mathsf{S}'})$. For a given the the target sparsity k, we have that:

$$\mathsf{s}_* = \arg\max_{|\mathsf{s}|=k} J(\mathsf{s}). \tag{3.7}$$

3.3.1 Submodularity and Efficient Inference

In this section, we show that the cost function $J(\mathsf{s})$ is monotone submodular, and describe the greedy forward selection algorithm for efficient inference. Let the ground set $\mathsf{m} = [d]$, and note that $J(\mathsf{m}) = 0$ and $J(\emptyset) = \log p(\boldsymbol{x} = \boldsymbol{c})$, so $J(\emptyset) \leq J(s) \leq 0$. In parts of the discussion, it will be useful to normalize the cost function by shifting its value i.e. $\tilde{J}(\mathsf{s}) = J(\mathsf{s}) - J(\emptyset)$, so $0 \leq \tilde{J}(\mathsf{s}) \leq -J(\emptyset)$.

The following theorem explores the submodularity of subset selection using relative entropy.

Theorem 3.3.1 (Madiman and Tetali [2010])**.** *Let $q \in \mathcal{P}$ and $p \in \mathcal{P}$ be probability densities on $\tilde{\mathsf{X}}^d$, and let q_{s} and p_{s} be their marginals on $\tilde{\mathsf{X}}^{|\mathsf{s}|}$, such that the set function $F(\mathsf{s}) : \wp(d) \mapsto [0, \infty]$, $F(\mathsf{s}) = -\mathrm{KL}(q_{\mathsf{s}} \| p_{\mathsf{s}})$ does not take the value $-\infty$ for any $\mathsf{s} \in \wp(d)$, then $F(\mathsf{s})$ is submodular.*

We now consider the submodularity of $J(\mathsf{s})$.

Theorem 3.3.2. *$J(\mathsf{s})$ is monotone submodular.*

Proof. Monotone: Let $\mathsf{c} \subset \mathsf{s}$, then:

$$p(\boldsymbol{x}_{\mathsf{M}\setminus\mathsf{c}}) = p(\boldsymbol{x}_{\mathsf{M}\setminus\mathsf{s}}, \boldsymbol{x}_{\mathsf{M}\setminus\mathsf{s}\cap\mathsf{c}}) \leq p(\boldsymbol{x}_{\mathsf{M}\setminus\mathsf{s}}).$$

Submodular: Consider $F(\mathsf{s}) = \log p(\boldsymbol{x}_{\mathsf{s}} = \boldsymbol{c}_{\mathsf{s}})$. Recall that $\mathrm{KL}(\delta_{\mathsf{s}} \| p_{\mathsf{s}}) = -\log p(\boldsymbol{x}_{\mathsf{s}} = \boldsymbol{c}_{\mathsf{s}})$ [Williams, 1980], and that $F(\mathsf{s})$ is bounded above and below. Thus, $F(\mathsf{s})$ is submodular by Theorem 3.3.1. Finally, we note that if $F(\mathsf{s})$ is submodular, so is its reflection $J(\mathsf{s}) = F(\mathsf{m}\setminus\mathsf{s})$. □

While maximization of submodular functions is generally NP-hard, a simple greedy forward selection heuristic (Algorithm 2), has been shown to perform almost as well as the optimal in practice, and is known to have strong theoretical guarantees.

Theorem 3.3.3 (Nemhauser et al. [1978])**.** *In the case of any normalized, monotonic submodular function F, the set s_* obtained by the greedy algorithm achieves at least a constant fraction $\left(1 - \frac{1}{e}\right)$ of the objective value obtained by the optimal solution i.e.*

$$F(\mathsf{s}_*) = \left(1 - \frac{1}{e}\right) \max_{|\mathsf{s}| \leq k} F(\mathsf{s}).$$

In addition, no polynomial time algorithm can provide a better approximation guarantee unless P = NP [Feige, 1998]. For the special case when p is product form, we can give stronger guarantees.

Algorithm 2 Greedy selection $\max J(\mathsf{s})$ s.t. $|\mathsf{s}| = k$

Input: $k, \mathsf{s} = \emptyset$
while $|\mathsf{s}| < k$ **do**
  foreach $i \in \mathsf{m} \backslash \mathsf{s}$, $f_i = J(\mathsf{s} \cup i) - J(\mathsf{s})$
  $\mathsf{s} = \mathsf{s} \cup \{\arg\max f_i\}$
end while
Return: s.

Corollary 3.3.4. *Let $J(\mathsf{s})$ be defined as in Theorem 3.3.2 and suppose the base density is product form i.e. $p(\boldsymbol{x}) = \prod_{i=1}^{d} p(x_i)$, then $J(\mathsf{s})$ is linear.*

Proof. Define $\mathbf{1}_{\mathsf{s}} \in \mathfrak{R}^d$ as the vector $(\mathbf{1}_{\mathsf{s}})_i = 1$ if $i \in \mathsf{s}$ and zero otherwise, and define the vector $\boldsymbol{h} \in \mathfrak{R}^d$ taking values $\boldsymbol{h}_i = p(x_i = c_i)$. When $p(\boldsymbol{x}) = \prod_{i=1}^{d} p(x_i)$, we have that $J(s) = \log p(\boldsymbol{x}_{\mathsf{s}'} = \boldsymbol{c}_{\mathsf{s}'}) = \sum_{i \in \mathsf{s}'} \log p(x_i) = \langle \mathbf{1}_{\mathsf{s}'}, \boldsymbol{h} \rangle$, a linear function. □

An additional benefit of the greedy approach is that it does not require the decision of the support size k to be made at training time. As an *anytime* algorithm, training can be stopped at any k based on computational constraints, while still returning meaningful results. We present empirical evaluation to the above approach in Sections 3.4.1 for sparse probabilistic regression and compare against several established baselines.

3.3.2 General Structured Sparsity Constraints

Structured sparsity extends classic sparsity constraints with additional information on the sparse subsets. For example, the sparsity could be constrained by a tree structure so that selection of a parent node implicitly se-

lects all its children as well. The structural constraint can be encoded as a matroid (N, E) where N are the base set of dimensions, and E represents the set of all possible candidate solutions under the given constraint. General structured sparsity is challenging to model using standard prior design techniques. Instead, one may consider Bayesian inference using the structured prior distribution recovered by restricting the base prior to the union of all possible structured subsets. As in the classic sparsity case, we consider an approximation of the resulting posterior based on the variable set which captures the maximum posterior mass. The resulting (N, E)-matroid constrained information projection of a density p is simply given by:

$$\min_{\mathsf{S} \in \mathsf{E}} \min_{\mathrm{supp}(q) \in \mathsf{S}} \mathrm{KL}(q \| p). \tag{3.8}$$

A simple greedy algorithm on the enumeration of the matroid as outlined in Algorithm 3 can be used for support selection under general matroid constraints. Note that the greedy selection algorithm for the classic sparsity case is a special case of Algorithm 3 with a uniform matroid. For the more general matroid constraints, greedy selection on the enumeration admits slightly weaker guarantees. Improved approximation guarantees can be achieved by randomized algorithms [Calinescu et al., 2011].

Theorem 3.3.5 (Calinescu et al. [2011])**.** *Algorithm 3 guarantees a constant factor approximation of* $1/2$ *for* (3.8)*.*

Multi-view sparsity. A special case of structured sparsity is the multi-view sparsity. The base set of dimensions are divided into v views/groups. Also given is a set of maximum number of allowed selections from each view $\{k_1, k_2, \ldots, k_v\}$. In other words, no more than k_i selections can be made from the i^{th} view/group. It should be straightforward to see that the multi-view sparsity constraint induces a partition matroid structure, and as such Algorithm 3 is applicable. Algorithm 3 can be easily re-written for the partition sparsity constraint to avoid exhaustive enumeration of the set E as Algorithm 4. The 1/2 factor approximation guarantee carries over for Algorithm 4. We shall see in the sequel that this particular algorithm leads to an efficient inference algorithm for sparse Probabilistic CCA.

Group Sparsity Constraints. Group sparsity involves selecting variables from r groups subject to the constraint that if a group is selected, all the variables within the group must be selected, but no more than k variables can be selected in all. Let $\mathsf{G} = \{\mathsf{G}_1, \mathsf{G}_2, \ldots, \mathsf{G}_r\}$ represent the set of r groups, so that $\forall i, \mathsf{G}_i \subset [d]$ and $\forall i \neq j, \mathsf{G}_i \cap \mathsf{G}_j = \emptyset$. As in the classic sparse case, information projection to the set of all group sparse subsets of $[d]$ is intractable in general. Instead, we propose approximate inference by seeking the projection to the set which maximizes the captured mass of p. The resulting group sparsity constrained information projection of a density p is given by:

$$\min_{\mathsf{S} \subset [r]} \min_{\left\{q \,|\, supp(q) \subset \bigcup_{i \in \mathsf{S}} \mathsf{G}_i, \sum_{i \in \mathsf{S}} |\mathsf{G}_i| \leq k\right\}} \mathrm{KL}(q \| p). \tag{3.9}$$

Theorem 3.3.6. *The group selection problem* (3.9) *is equivalent to a normalized monotone submodular maximization problem with a knapsack constraint.*

Proof. We prove by mapping (3.9) to an equivalent problem by performing a variable change.

Let $\mathsf{G_S} := \bigcup_{i \in \mathsf{S}} \mathsf{G}_i$. From Section 3.1, the inner optimization is $\min_{|\mathsf{G_S}| \leq k, q \in \mathcal{F}_{\mathsf{G_S}}} \mathrm{KL(q\|p)} = -\log p(\mathbf{x}_{\mathsf{G} \setminus \mathsf{G_S}})$

Define the function $J : \wp(r) \to \mathbb{R}$ as $J(\mathsf{S}) := \log \mathrm{p}(\mathbf{x}_{\mathsf{G} \setminus \mathsf{G_S}} = 0)$, and the function $\tilde{J} : \wp(r) \to \mathbb{R}$ as $\tilde{J}(\mathsf{S}) := J(\mathsf{S}) - J(\emptyset)$.

Define the costs associated with picking G_i as $c_i = |\mathsf{G}_i| \forall i \in [r]$. The cost function of a set $\mathsf{s} \subset \mathsf{G}$ can thus be written as $c(\mathsf{s}) := \sum_{\forall i \text{ s.t. } \mathsf{G}_i \in \mathsf{s}} c_i$ The optimization problem 3.9 is then equivalent to $\max_{\sum_{i \in \mathsf{S}} c_i \leq k} \tilde{J}(\mathsf{S})$.

The result follows from Theorem 3.3.2.

□

We now present a re-weighted greedy algorithm with partial enumeration in Algorithm 5 to solve (3.9). The re-weighting is ensures that the greedy step chooses the best possible myopic marginal gain. However, with the re-weighting alone the approximation factor can be arbitrarily bad. To bound it to a constant factor, partial enumeration is required. We also note that Algorithm 5 is *not* a special case of Algorithm 3, as it exploits the special structure of group sparsity to construct a scheme with improved optimization-theoretic

guarantees. The following theorem establishes the optimization guarantee of Algorithm 5.

Theorem 3.3.7 (Sviridenko [2004])**.** *Algorithm 5 with* $m = 3$ *guarantees a constant factor approximation of* $(1 - \frac{1}{e})$ *for* (3.9)*.*

3.3.3 Gaussian Base Measure and Support Constraints

We consider a special case of approximate sparse support inference where $\mathsf{X} = \mathfrak{R}^D$, and the base measure is a Gaussian distribution. Let $\boldsymbol{\beta} \sim \boldsymbol{P}$ be drawn from a *multivariate Gaussian* distribution. The density is given as:

$$\mathcal{N}(\boldsymbol{m}, \mathbf{S}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{m})^{\dagger}\mathbf{S}^{-1}(\boldsymbol{\beta} - \boldsymbol{m})\right)}{(2\pi)^{d/2}|\mathbf{S}|^{1/2}},$$

where $\boldsymbol{m} \in \mathfrak{R}^d$ is the mean vector and $\mathbf{S} \in \mathfrak{R}^{d \times d}$ is the covariance matrix. We set the default value $c_i = 0 \; \forall \; i \in [d]$.

Algorithm 3 GreedyMatroid(N, E)

Input: Matroid (N, E)
$\mathsf{A} \leftarrow \emptyset$
while N is not empty **do**
  $s^* \leftarrow \arg\max_{s \in \mathsf{N}} J(\mathsf{A} \cup \{s\}) - J(\mathsf{A})$
  if $\mathsf{A} \cup \{s^*\} \in \mathsf{E}$ **then**
    $\mathsf{A} = \mathsf{A} \cup \{s^*\}$
  end if
  $\mathsf{N} = \mathsf{N} - \{s^*\}$
end while
Return A

Algorithm 4 GreedyMultiView ($k_1, \ldots, k_v$, $m(\cdot)$)

Input : N, Sparsities $\{k_1, k_2, \ldots, k_v\}$, mapping function $m : [d] \to [v]$.
$\mathsf{A} \leftarrow \emptyset$
selected[i]=0, $\forall i \in [v]$
while N is not empty **do**
  $s^* \leftarrow \arg\max_{s \in \mathsf{N}} J(\mathsf{A} \cup \{s\}) - J(\mathsf{A})$
  if selected$[m(s^*)] < k_i$ **then**
    $\mathsf{A} = \mathsf{A} \cup \{s^*\}$
    selected$[m(s^*)]$ +=1
  end if
  $\mathsf{N} = \mathsf{N} - \{s^*\}$
end while
Return A

Algorithm 5 GreedyPartialEnum (G, k, $c(\cdot)$)

Input: Set of groups G, Total max sparsity k, parameter m, cost function $c(\cdot)$
$\mathsf{S}_1 \leftarrow \arg\max_{\mathsf{s} \subset \mathsf{G}, |\mathsf{s}| < m, c(\mathsf{s}) \leq k} \tilde{J}(\mathsf{s})$
$\mathsf{S}_2 \leftarrow \emptyset$
for all $\mathsf{s} \subset \mathsf{G}, |\mathsf{s}| = m, c(\mathsf{s}) \leq k$ **do**
  $\mathsf{S}_3 \leftarrow$ ReweightedGreedy(G, $k - m - 1$, $c(\cdot)$, s)
  if $\tilde{J}(\mathsf{S}_2) \leq \tilde{J}(\mathsf{S}_3)$ **then**
    $\mathsf{S}_2 \leftarrow \mathsf{S}_3$
  end if
end for
Return $\arg\max\{\tilde{J}(\mathsf{S}_1), \tilde{J}(\mathsf{S}_2)\}$

Algorithm 6 ReweightedGreedy ($\bar{\mathsf{G}}$, $\bar{k}$, $c(\cdot)$, $\bar{\mathsf{S}}_2$)

Input: Set of groups $\bar{\mathsf{G}}$, Total max sparsity $\bar{k}$, cost function $c(\cdot)$, Init groups $\bar{\mathsf{S}}_2$
$\mathsf{A} \leftarrow \bar{\mathsf{S}}_2$
while $\bar{\mathsf{G}} \backslash \mathsf{A} \neq \emptyset$ **do**
  $\mathsf{s}^* \leftarrow \max_{\mathsf{s} \in \bar{\mathsf{G}} \backslash \mathsf{A}} \frac{J(\mathsf{A} \cup \mathsf{s}) - J(\mathsf{A})}{c(\mathsf{s})}$
  if $c(\mathsf{A} \cup \mathsf{s}^*) \leq k$ **then**
    $\mathsf{A} = \mathsf{A} \cup \mathsf{s}^*$
  end if
  $\bar{\mathsf{G}} = \bar{\mathsf{G}} - \mathsf{s}^*$
end while
Return A

Consider a generative model for $[n] \ni i$ samples given by a linear model combined with Gaussian noise $y_i|_{\boldsymbol{\beta}} = \boldsymbol{\beta}^\dagger \boldsymbol{z}_i + \epsilon$, where the response $y_i \in \mathfrak{R}$, the feature vector $\boldsymbol{z}_i \in \mathfrak{R}^d$, the weight vector $\boldsymbol{\beta} \in \mathfrak{R}^d$. The weights are drawn from the zero mean Gaussian distribution $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$, where $\mathbf{C} \in \mathfrak{R}^{d \times d}$ is the prior covariance matrix, and its inverse $\mathbf{D} = \mathbf{C}^{-1} \in \mathfrak{R}^{d \times d}$ is the corresponding prior precision matrix. The noise is drawn from a univariate Gaussian $\epsilon \in \mathfrak{R}$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We set $\lambda = \frac{1}{\sigma^2}$. Let $\boldsymbol{y} \in \mathfrak{R}^n$ represent the responses collected into a vector with $\boldsymbol{y}(i) = y_i$, and $\mathbf{Z} \in \mathfrak{R}^{n \times d}$ represent the features in matrix form, so $\mathbf{Z}(i, :) = \boldsymbol{z}_i^\dagger$.

As the prior and the likelihood are Gaussian, the unconstrained posterior distribution $\boldsymbol{P}(\boldsymbol{\beta}|\boldsymbol{y})$ is Gaussian, represented as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} \in \mathfrak{R}^{d \times d}$ is the posterior covariance matrix, and its inverse $\boldsymbol{\Lambda} = \mathbf{S}^{-1} \in \mathfrak{R}^{d \times d}$ is the corresponding precision matrix. The posterior precision is given by $\boldsymbol{\Lambda} = \mathbf{D} + \lambda \mathbf{Z}^\dagger \mathbf{Z}$. The posterior mean $\boldsymbol{\mu} \in \mathfrak{R}^d$ is given by $\boldsymbol{\mu} = \lambda \boldsymbol{\Lambda} \mathbf{Z}^\dagger \mathbf{y}$. Recall that $\boldsymbol{\mu}_{\mathsf{s}} \in \mathfrak{R}^{|\mathsf{s}|}$ is the subvector given by $\boldsymbol{\mu}_{\mathsf{s}} = \{\boldsymbol{\mu}(i) \mid i \in \mathsf{s}\}$. For matrices, define $\boldsymbol{\Sigma}_{\mathsf{s},\mathsf{c}} \in \mathfrak{R}^{|\mathsf{s}| \times |\mathsf{c}|}$ as the submatrix $\{\boldsymbol{\Sigma}(i, , j) \mid i \in \mathsf{s}, j \in \mathsf{c}\}$. We also define the linear projection matrix $\mathbf{P}_{\mathsf{s}} \in \mathfrak{R}^{d \times |\mathsf{s}|}$, $\mathbf{P}_{\mathsf{s}} : \mathfrak{R}^{|\mathsf{s}|} \mapsto \mathfrak{R}^d$ by imputing zeros as missing entries.

For any fixed s, the information projection is given by the restriction of $\boldsymbol{P}(\boldsymbol{\beta}|\mathbf{y})$ as the conditional distribution:

$$\boldsymbol{P}(\boldsymbol{\beta}_{\mathsf{s}}|\boldsymbol{\beta}_{\mathsf{s}'} = \mathbf{0}, \mathbf{y}) = \mathcal{N}(\boldsymbol{m}_{\mathsf{s}|\mathsf{s}'}, \boldsymbol{\Sigma}_{\mathsf{s}|\mathsf{s}'}), \tag{3.10}$$

where $\boldsymbol{m}_{\mathsf{s}|\mathsf{s}'} \in \mathfrak{R}^{|\mathsf{s}|}$, given by $\boldsymbol{m}_{\mathsf{s}|\mathsf{s}'} = \boldsymbol{\mu}_{\mathsf{s}} + \boldsymbol{\Lambda}_{\mathsf{s},\mathsf{s}}^{-1} \boldsymbol{\Lambda}_{\mathsf{s},\mathsf{s}'} \boldsymbol{\mu}_{\mathsf{s}}$. Approximate inference is applied as shown in (3.5) using the convex sparse subsets, equivalent to the

submodular optimization (3.7). The resulting cost function (up to constants) is given by $J(\mathsf{s})$:

$$= \boldsymbol{\mu}_{\mathsf{s}'}^{\dagger} \boldsymbol{\Sigma}_{\mathsf{s}',\mathsf{s}'}^{-1} \boldsymbol{\mu}_{\mathsf{s}'} - \log |\boldsymbol{\Sigma}_{\mathsf{s}',\mathsf{s}'}| + a_1 \tag{3.11}$$

$$= (\boldsymbol{\mu} - \mathbf{P}_{\mathsf{s}} \boldsymbol{m}_{\mathsf{s}|\mathsf{s}'})^{\dagger} \boldsymbol{\Lambda} (\boldsymbol{\mu} - \mathbf{P}_{\mathsf{s}} \boldsymbol{m}_{\mathsf{s}|\mathsf{s}'}) - \log |\boldsymbol{\Lambda}_{\mathsf{s},\mathsf{s}}|, \tag{3.12}$$

where a_1 is an additive constant. (3.11) is follows directly from the cost function (3.7), while (3.12) is derived most easily by directly solving the information projection (3.5) for a fixed s. (3.12) also simplifies the interpretation of the cost function. The subsets are selected in order to minimize the distance between the conditional mean and marginal mean vector, additionally the determinant term measures the coupling between the variables.

3.3.4 Group Sparse Linear Regression

Consider a generative model for n samples given by a linear model and an additive Gaussian noise: $\mathbf{y} = \mathbf{Z}\boldsymbol{\beta} + \epsilon,$, where $y \in \mathbb{R}^n$ is the response, $\mathbf{Z} \in \mathbb{R}^{n \times d}$ is the feature matrix, and $\boldsymbol{\beta} \in \mathbb{R}^d$ is the vector of regression weights. The weights have an associated normal prior, $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$ for a known $\mathbf{C} \in \mathbb{R}^{d \times d}$. The noise ϵ is drawn from a Gaussian $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The posterior distribution of β is also a Gaussian, $\mathrm{p}(\boldsymbol{\beta}|\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and can be written in closed form by standard Bayes theorem with $\boldsymbol{\Sigma}^{-1} = \mathbf{C}^{-1} + \frac{1}{\sigma^2}\mathbf{Z}^{\top}\mathbf{Z}$, and, $\boldsymbol{\mu} = \frac{1}{\sigma^2}\boldsymbol{\Sigma}\mathbf{Z}^{\top}\mathbf{y}$.

Let $\mathsf{G} = \{\mathsf{G}_1, \mathsf{G}_2, \ldots, \mathsf{G}_r\}$ be the given set of groups so that $\forall i \in [r], \mathsf{G}_i \subset [d]$, and $\forall i \neq j, \mathsf{G}_i \cap \mathsf{G}_j = \emptyset$. The optimization problem for sparse group selection is then given by (3.9). For the spacial case where p is Gaussian,

the information projection to any structured subset remains in the Gaussian family [Koyejo and Ghosh, 2013]. Thus, the search for q in (3.9) can be restricted to Gaussians. Define $\mathbf{r} = \frac{1}{\sigma^2}\mathbf{Z}^\top \mathbf{y}$. It is easy to show by expanding the KL that (3.9) for group sparse linear regression is equivalent to the submodular maximization problem:

$$\max_{\left\{\mathsf{S}\subset[r],\, \mathsf{S}=\bigcup_{i\in\mathsf{S}}\mathsf{G}_i,\, |\mathsf{S}|\leq k\right\}} \mathbf{r}_{\mathsf{s}}^\top [\boldsymbol{\Sigma}^{-1}]_{\mathsf{s}} \mathbf{r}_{\mathsf{s}} - \log\det[\boldsymbol{\Sigma}^{-1}]_{\mathsf{s}}. \tag{3.13}$$

Once the support s is selected, the respective approximate posterior q^* can be obtained as the respective conditional $\mathrm{q}^*(\mathbf{x}) = \mathrm{p}(\mathbf{x}|\mathbf{x}_{\mathsf{S}^c} = 0)$.

3.4 Experiments

In this section, we present experimental results comparing the sparse approximate inference approach to other sparsity inducing Bayesian and frequentist models. We begin by first validating our framework using simulated data experiments, where the data generating process is known and controlled.

3.4.1 Sparse Linear Regression

We employ the special case of our framework with cardinality constraints (Section 3.3.1) and Gaussian priors(Section 3.3.3). The compared models are as follows:

- **Regularized least squares** (*Ridge*): The standard regularized least squares estimator - analogous to the full posterior distribution of the Gaussian model outlined in Section 3.3.1 assuming a univariate Gaussian

prior. While *Ridge* does not return sparse weights, it typically performs well in terms of predictive accuracy.

- **Least absolute shrinkage and selection Operator** (*Lasso*) [Tibshirani, 1996]: A popular frequentist model for estimating sparse regression weights. It combines a least squares loss with a sum of absolute value regularization.

- **Automatic relevance determination** (*ARD*) [Wipf and Nagarajan, 2007]: This is a maximum likelihood approach for estimating the diagonal prior covariance of a Gaussian model. The estimated covariance values are often sparse in practice, and combined with zero prior mean, leads to sparse distributions and a sparse mean weight weight vector. Like *Sparse-G.* , *ARD* estimates the full posterior distribution of the parameters.

- **Spike and slab prior** [Mitchell and Beauchamp, 1988] estimates the weight vectors using a mixture of a delta function at zero (Spike), and a Gaussian distribution with non-zero mean (Slab) for each dimension. As the with most sparse Bayesian models, the posterior mean of the estimated weight vectors is not sparse. Furthermore, fitting the exact model is tedious. So we compare against several versions - (i) SpikeSlabFull runs the full model and truncates away support dimensions with posterior probability less than 0.5, (ii) SpikeSlabVar fits a variational approximation [Ishwaran and Rao, 2005] which is faster to fit and more scalable,

(iii) SpikeSlabVar0.5 fits SpikeSlabVar and truncates away dimensions with posterior probability ¡ 0.5, (iv) SpikeSlabVarKL fits SpikeSlabVar and then perform a greedy KL projection onto top k dimensions. Note that the support of SpikeSlabVar is not sparse.

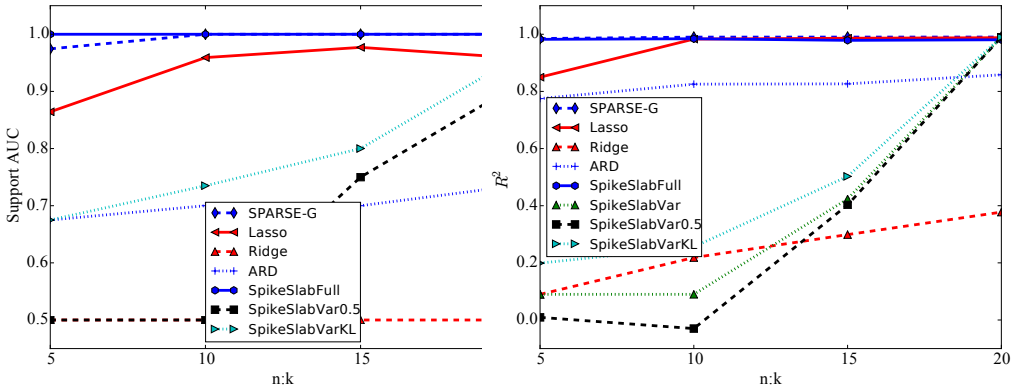
Approximate inference for Sparse-G was performed using the greedy forward subset selection approach outlined in Algorithm 2. Our implementation avoids large matrix inverses and uses the fact that the determinant can be computed incrementally using the Schur complement lemma, and the quadratic terms can be computed incrementally using the Woodburry matrix identity. Further details are left for a longer version of this manuscript. We implemented *ARD* using iterative re-weighted Lasso as suggested by Wipf and Nagarajan [2007]. *Ridge* and *Lasso* were optimized using implementations from the *scikit-learn* python package [Pedregosa et al., 2011]. For each of these models, we performed hyperparameter selection using inner loop 5-fold cross validation.

We are most interested in high dimensional regression where the model parameters are generated from a distribution supported on a sparse domain, and in data sets with more dimensions than samples as found in high dimensional scientific data. In such scenarios, the model constraints can be critical for effective regression and parameter estimation. We performed experiments using simulated data that matches these characteristics. We tested the models ability to estimate the support and values of the weight vector, and the predictive accuracy of the reconstructed targets.

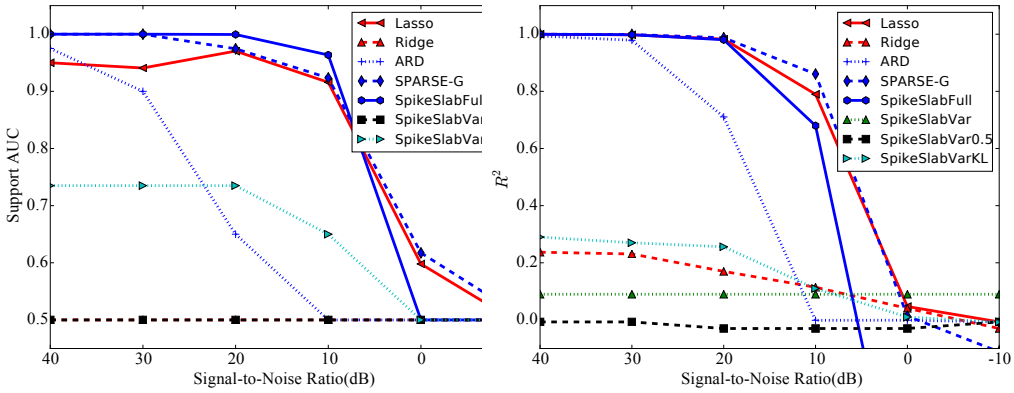
The regression accuracy was measured using the coefficient of determination on the test set. The R^2 metric given by $1 - \sum(\hat{y} - y)^2 / \sum(y - \bar{y})^2$ where y is the target response with sample mean $\bar{y}$ and $\hat{y}$ is the predicted response. R^2 measures the gain in predictive accuracy compared to a mean model and has a maximum value of 1. The support recovery was measured using the AUC of the recovered support with respect to s_*.

We generated random high dimensional feature vectors $a_i \in \mathfrak{R}^d$ with $a_i \sim \mathcal{N}(0, 1)$. The response was generated as $y_i = w^\top a_i + \nu_i$ where ν_i represents independent additive noise with $\nu_i \sim \mathcal{N}(0, \sigma^2)$ for all $i \in [n]$. We set σ^2 implicitly via the signal to noise ration (SNR) as SNR $= \text{var}(y)/\sigma^2$, where $\text{var}(y)$ is the variance of y. In each experiment, we sampled a sparse weight vector w by sampling k dimensions at random with from $[d]$, then we sampled values $w_i \sim \mathcal{N}(0, 1)$ and set other dimensions to zero. We performed a series of tests to investigate the performance of the model in different scenarios. Each experiment was run 10 times with separate training and test sets. We present the average results on the test set.

Our first experiment tested the performance of all models with limited samples. Here we set $k = 20, d = 1000$ and an SNR of 20dB. The number of training values was varied from $n = 100, \ldots, 400$ with 200 test samples. Fig. 3.3a shows the model performance in terms of support recovery. With limited training samples, Sparse-G outperformed all the baselines except SpikeSlabFull, and performs as well as SpikeSlabFull. We also found that SpikeSlabVarKL consistently outperformed SpikeSlabVar0.5. We specu-



(a) AUC as a function of $n : k$ ratio (b) R^2 as a function of $n : k$ ratio



(c) AUC as a function of SNR (d) R^2 as a function of SNR

Figure 3.3: Performance of Sparse Linear Regression approaches on simulated data (Section 3.4.1)

late that the significant gap between Sparse-G and SpikeSlabVarKL may be partly due to the mean field assumption. Fig. 3.3b shows the corresponding regression performance.

Our second experiment tested the performance of all models with high levels of noise. Here we set$k = 20, d = 1000$ and $n = 200$ with 200 test samples. We varied the SNR from 40dB to 10dB (note that σ^2 increases as SNR is decreased). Fig. 3.3c shows the support recovery performance of the different models. We found a performance gap between Sparse-G and Lasso, more pronounced than in the small SNR test. The SpikeSlabVar0.5 was the worst performing model, but the performance was improved by SpikeSlabVarKL. Only Sparse-G and SpikeSlabFull achieved perfect support recovery at low noise (high SNR) levels. The regression performance is shown in Fig. 3.3d. While ARD and Lasso matched Sparse-G at low noise levels (high SNR), their performance degraded much faster at higher noise levels (low SNR).

For both of the above experiments, the exact Spike-and-Slab takes a long time to converge, so we have only used the mean field variational approximation. To illustrate and compare against the exact Spike and Slab, we repeat the above two experiments with $d = 1000$, with everything else in the experimental setup remaining the same. The results are shown in Figure 3.3. The exact Spike and Slab consistently performs well for support recovery and test R^2 in cases when the SNR is relatively low, while Sparse-G remains competitive. However, for cases with moderate to high SNR, the Spike and Slab tends to overfit, and performs badly with test $R^2 < 0$. On the other hand, Sparse-G

	Time Taken (sec)	
# Dimensions	SPARSE-G	Spike and Slab
100	0.22	1.43
500	1.04	88.23
1000	2.37	450.58
5000	9.45	62.8k

Table 3.1: Time taken for Sparse-G vs exact Spike and Slab

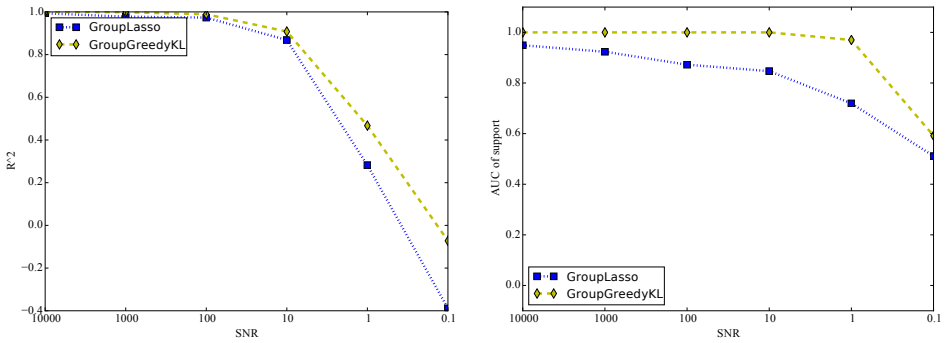
maintains a more graceful degradation with increasing SNR and decreasing training size represented by $n : k$.

Timing Experiments: In this section, we study the time taken to train by Sparse-G vs time taken to train using exact Spike and Slab and show that Sparse-G is the much faster especially for higher dimensional cases. We use the simulated data process as detailed in the beginning of this section with the lowest SNR. We generate different data sets with $d = 100, 500, 1000, 5000$ with all other settings remaining the same. We note the time taken for both exact Spike and Slab, and Sparse-G to run to completion with training data size $n = 200$. The results are presented in Table 3.1.

3.4.2 Bayesian Regression with Group Sparsity

We compare the proposed approach for group sparsity (Section 3.3.4) against the sparse-group lasso [Simon et al., 2013] implemented in the package SLEP [Liu et al., 2009] which is used in practice as state of the art. We fix the ambient dimension to be $d = 1000$. We generate an arbitrary fixed weight vector $\boldsymbol{\beta} \in \mathfrak{R}^d$ with all but $k = 20$ dimensions zeroed out, arbitrar-

ily separated into 5 groups of 4 each. We sample from the d-variate normal distribution with identity covariance $n = 1000$ times to get the feature matrix $\mathbf{X} \in \mathfrak{R}^{n \times d}$. Finally we obtain the response vector $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with σ^2 being set with varying values of the Signal-to-Noise ratio (SNR) so that SNR=$\{10000, 1000, 100, 10, 1, 0.1\}$ to generate 6 data sets. Note that SNR < 1 implies variance of the noise is more than that of the signal. We split the data $50 - 10 - 40$ into training, validation and test sets. We compare performance of GroupGreedyKL (group selection based on KL projection) and GroupLasso [Simon et al., 2013] on two metrics - the AUC of the support recovered, and R^2 on test data. We use Bayes Factor to estimate k for GroupGreedyKL. For GroupLasso, we do a parameter sweep to get the best performing numbers. For each of the 6 different SNRs, data is generated 10 different times randomly and the average results are reported. The results are presented in Figure 3.4. GroupGreedyKL performs consistently better than GroupLasso, and degrades more gracefully as SNR decreases.



(a) R^2 performance on test data (b) Recovery of true support

Figure 3.4: Group Sparse Regression performance on simulated data.

Chapter 4

Sparse Probabilistic Factor Models

In this chapter, we present algorithms to apply the sparse prior construction methodology developed in Chapter 3 to structured sparse probabilistic factor models. We consider our data to consist of n observations of vector valued variables in d dimensional ambient space, which are stacked in a matrix $\mathbf{T} \in \mathbb{R}^{n \times d}$. Drawing inspiration from traditional PCA, we seek a few sparse basis vectors whose linear combination generates the observation matrix with small error. The observation matrix $\mathbf{T}$ is modelled as a product of a parameter $\mathbf{X} \in \mathfrak{R}^{n \times r}$ and a *sparse* $\mathbf{W} \in \mathfrak{R}^{r \times d}$. Thus, the sparse basis vectors are stacked as rows of $\mathbf{W}$, and their linear combination is modeled by $\mathbf{X}$. In cases which have $n \gg d$, the above factorization is useful for small r, which is set according to the domain. Let $\boldsymbol{\mu}$ be the matrix of column means of $\mathbf{T}$ generated as, $\boldsymbol{\mu} = \text{columnMeans}(\mathbf{T})^{\dagger} \otimes \mathbf{1}$, and Gaussian noise is represented by $\boldsymbol{\epsilon}_{ij} \sim \text{N}(0, \sigma^2), \forall i \in [n], \forall j \in [d]$.

The observation model is then represented as:

$$\mathbf{T} = \mathbf{XW} + \boldsymbol{\mu} + \boldsymbol{\epsilon}.$$

We use a normal prior for each row of $\mathbf{W}$ i.e. $\mathbf{W}_{i,\cdot} \sim \text{N}(\mathbf{0}, \mathbf{C})\ \forall i \in [r]$, with a given a prior covariance matrix $\mathbf{C}$. We assume individual rows of $\mathbf{W}$

are independent. The joint distribution can also be written as the matrix-variate normal $\mathbf{W} \sim \text{MVN}(0, \mathbf{C}, \mathbf{I})$. In the proposed model, the parameters are given by $\mathbf{\Theta} = \{\mathbf{X}, \sigma^2\}$, while $\mathbf{W}$ are the latent variables. Inference and learning can be performed using the EM algorithm (Section 4.1) applied on the log-likelihood $\log p(\mathbf{T}; \mathbf{X}, \sigma^2)$. As we shall see, the application of structural constraints of sparsity on the factors leads to a variational E-step.

We show that our framework performs very competitively when compared to established baselines on simulated and real world functional magnetic resonance imaging (fMRI) data, often beating the baselines by orders of magnitude when compared on the standard metrics.

Neuroimaging applications are well-suited to test our framework for many reasons. FMRI data sets are typically high dimensional, in order of 100,000 dimensions and a few hundred samples. In such cases, as we shall soon see, many traditional models do not perform well. Furthermore, fMRI applications involve a variety of tasks based on the respective experiment design that require different machine learning models and algorithms to aid in uncovering the underlying structural properties. This allows us to test several applications of our frameworks for different tasks. Finally, many past studies provide us with easy data collection and preprocessing schemes. For different tasks and data sets, we show that our framework achieves state of the art performance for:

- Sparse Probabilistic PCA on the fMRI Resting State data, with empiri-

cal comparison to Generalized Power Method, Truncated Power Method, Low Rank PCA, Path Sparse PCA, Online Dictionary Learning. (Section 4.5.1.1).

- Group Sparse Probabilistic PCA on the fMRI Neurovault data with empirical comparison to the Structured Sparse PCA algorithm of Jenatton et al. [2010] (Section 4.5.1.2).

- Group Sparse Collective Matrix Factorization on the Human Connectome data with empirical comparison to sparse Canonical Correlation Analysis (Section 4.5.1.3).

All of the above fMRI data sets are high-dimensional, with the number of dimensions far exceeding the number of samples. We also provide qualitative analysis of the extracted brain voxels in the above experiments from a neuroscience perspective.

4.1 Inference with Sparse Constraints

In this section, we illustrate how priors constructed by restricting domain to sparse supports can be incorporated in practical algorithms for sparse inference.

Expectation Maximization can be described using the free energy interpretation [Neal and Hinton, 1998]. Maximizing the negative log-likelihood can be shown to be equivalent to maximizing a free energy function $\mathscr{F}$ (see (4.1)).

The E-step can be viewed as the search over the space of distributions q(.) of the latent variables $\mathbf{W}$, keeping the parameters Θ fixed (4.2), and the M-step can be interpreted to be the search over the parameter space, keeping the latent variables $\mathbf{W}$ fixed (4.3). The cost function for the EM is given by Neal and Hinton [1998]:

$$\mathscr{F}(q(\mathbf{W}), \Theta) = -\mathrm{KL}(q(\mathbf{W}) \| p(\mathbf{W}|\mathbf{T}; \Theta)) + \log p(\mathbf{T}; \Theta). \tag{4.1}$$

$$\text{E-step: } \max_{q} \mathscr{F}(q(\mathbf{W}), \Theta), \tag{4.2}$$

$$\text{M-step: } \max_{\Theta} \mathscr{F}(q(\mathbf{W}), \Theta). \tag{4.3}$$

This view of the EM algorithm provides the flexibility to design algorithms with any E and M steps that monotonically increase $\mathscr{F}$.

4.2 Variational E-step

An unconstrained optimization over q in (4.2) returns the posterior $p(\mathbf{W}|\mathbf{T}; \Theta)$. Variational methods perform the search for best q over a constrained set [Tzikas et al., 2008]. Let D be the set of distributions over $\mathbf{W}$ that fully factorize over individual rows of $\mathbf{W}$: $q(\mathbf{W}) = \prod_{i=1}^{r} q(\mathbf{W}_{i,.})$. We restrict the search over q to D. As a result of this restriction, we can optimize q($W_{i,.}$) for one i at a time in a co-ordinate descent fashion.

For introducing sparsity, we impose an additional constraint that $\forall i \in [r], q(\mathbf{W}_{i,\cdot})$ is k_i-sparse i.e. it has support only on at most k_i out of the ambient d dimensions. Let K_i be the set of all k_i-sparse supports. From (4.2) and (4.1), it follows that the variational E-step is minimization of KL divergence over the sets K_i. As shown in Section 3.1, information projection to a set is equivalent to restricting domain to the respective set. So, minimizing the KL-divergence can be thought as searching for the sparse support set that loses the least amount of information by restricting the domain set of distributions to sparse sets. We obtain an optimization problem over the constrained space of distributions:

$$\min_{\mathsf{K}_1} \dots \min_{\mathsf{K}_r} \min_{\substack{q(\mathbf{W}) \in \mathsf{D} \\ \forall i, \operatorname{Supp}(q(\mathbf{W}_{i,\cdot})) \in \mathsf{K}_i}} \mathrm{KL}(q(\mathbf{W}) \| p(\mathbf{W}|\mathbf{T}; \Theta)). \tag{4.4}$$

For Gaussian p, (4.4) can be re-written as an iterated information projection. For each row $i \in [r]$,

$$\min_{\mathsf{K}_i} \min_{\operatorname{Supp}(q(\mathbf{W}_{i,\cdot})) \in \mathsf{K}_i} \mathrm{KL}(q(\mathbf{W}_{i,\cdot}) \| \hat{p}(\mathbf{W}_{i,\cdot})), \tag{4.5}$$

where, with $\mathbf{W}_{\setminus i,\cdot}$ representing all rows of $\mathbf{W}$ except i, $\hat{p}_i$ depends on $q(\mathbf{W}_{\setminus i,\cdot})$ and $\log p(\mathbf{W}|\mathbf{T}; \Theta)$.

Thus, the independence assumption on $q(\cdot)$ allows for optimizing over each i individually, while holding the others fixed in a co-ordinate descent fashion. Each information projection monotonically decreases the free energy function.

Recall that the KL-gap for the constraining support was characterized in Section 3.1. We can simplify (4.5) for each i as

$$\text{For each row } i \in [r], \ \max_{\mathsf{K}_i} \log(\hat{p_i}([\mathbf{W}_{i,\cdot}]_{\mathsf{K}_i^c} = \mathbf{0}_{\mathsf{K}_i^c})). \tag{4.6}$$

(4.6) is the resulting discrete optimization problem to be solved for variational E-step for each i. By Theorem 3.3.2, optimization problem over the set of dimensions is submodular, and hence instead of exhaustive search, each of the k_i dimensions can be selected by a greedy algorithm which achieves at least a constant fraction $\left(1 - \frac{1}{e}\right)$ of the objective value obtained by the optimal solution [Nemhauser et al., 1978]. Moreover, no polynomial time algorithm can provide a better approximation guarantee unless P = NP Feige [1998].

To summarize, under the variational assumptions, the E-step can be solved iteratively over each i, and each optimization over i is a submodular discrete optimization problem with guaranteed constant factor approximation when using a greedy forward selection strategy. Moreover, since optimization over each i monotonically increases (or does not change) $\mathscr{F}$, updating the latent variable even for a single i in the E-step suffices. This is particularly helpful for PCA, as we see below when we derive explicit equations below.

Let $\mathbf{Z}_i = \mathbf{T} - \sum_{j \neq i} \mathbf{X}_{\cdot,j} \mathrm{E}\left[\mathbf{W}_{j,\cdot}\right]$. The optimization problem in the E-step is written as follows. For each row $i \in [r]$,

$$\max_{\mathsf{K}_i} \log(p([\mathbf{W}_{i,\cdot}]_{\mathsf{K}_i^c} = \mathbf{0}_{\mathsf{K}_i^c} | \mathbf{Z}_i; \mathbf{X}, \sigma^2)). \tag{4.7}$$

The posterior $p(\mathbf{W}_{i,\cdot})$ can be written as,

$$p(\mathbf{W}_{i,\cdot}|\mathbf{Z}_i; \mathbf{X}_{\cdot,i}, \sigma^2) \sim \mathcal{N}(\boldsymbol{m}^i, \boldsymbol{\Sigma}^i)$$

where,

$$[\boldsymbol{\Sigma}^i]^{-1} = \frac{1}{\sigma^2}(\mathbf{X}_{\cdot,i}^\dagger \mathbf{X}_{\cdot,i}) + \mathbf{C}^{-1},$$
$$\mathbf{m}^i = \boldsymbol{\Sigma}^i \ \frac{1}{\sigma^2}(\mathbf{X}_{\cdot,i}^\dagger)(\mathbf{Z}_i).$$

We can now expand (4.7) (see Section 3.3.3 for explicit derivation):

$$\max_{\mathsf{K}_i} \ \mathbf{r}_{\mathsf{K}_i}^{i\ \dagger}[[\boldsymbol{\Sigma}^{i-1}]_{\mathsf{K}_i}]^{-1}\mathbf{r}_{\mathsf{K}_i}^i - \log\det[\boldsymbol{\Sigma}^{i-1}]_{\mathsf{K}_i}, \quad \text{where } \ \mathbf{r}^i = \boldsymbol{\Sigma}^{i-1}\mathbf{m}^i. \tag{4.8}$$

Recall that $\boldsymbol{\Sigma}_{\mathsf{K}}$ is the submatrix of $\boldsymbol{\Sigma}$ supported on K, similarly for $[\boldsymbol{\Sigma}^{-1}]_{\mathsf{K}}$. After solving the constrained optimization problem (4.8) by a greedy selection for K_i^*, the resulting solution density, q_i^*, known to be the conditional by properties of the Gaussian, is given by:

$$q_i^* \sim \mathcal{N}(\mathbf{c}^i, \mathbf{D}^i)$$

where, (4.9)

$$[\mathbf{D}^i]^{-1} = [\boldsymbol{\Sigma}^{i-1}]_{\mathsf{K}_i^*}, \quad \mathbf{c}^i = \mathbf{D}^i\mathbf{r}_{\mathsf{K}_i^*}^i$$

Recall that q_i^* has support only on K_i^*, so in (4.9), $\mathbf{c}^i \in \mathfrak{R}^{|\mathsf{K}_i^*|}, \mathbf{D}^i \in \mathfrak{R}^{|\mathsf{K}_i^*|\times|\mathsf{K}_i^*|}$.

4.3 M-step

Since the free energy view of the EM shows that any M-step that increases $\mathscr{F}$ suffices, we maximize the log likelihood portion of $\mathscr{F}$ for the M-step.

It turns out solving for $\{\mathbf{X}, \sigma^2\}$ over $\mathscr{F}$ directly is computationally hard, so M-step is done for one column of $\mathbf{X}$ at a time, corresponding to row-wise E-step. If q^* is the distribution on $\mathbf{W}$ obtained from the E-step, the effective M-step for column i of $\mathbf{X}$ is:

$$\max_{\mathbf{X},\sigma^2} \mathbb{E}_{q^*} \left[\log p(\mathbf{Z}_i | \mathbf{W}_{i,\cdot}; \mathbf{X}_{\cdot,i}, \sigma^2)\right]. \tag{4.10}$$

For, any particular $i \in [d]$, let $\widehat{\mathbf{c}^i}$ represent the mean vector $\mathbf{c}^i$ expanded from $|\mathsf{K}_i^*|$ to ambient dimension d, with zeroes padded as needed. (4.10) can be re-written as:

$$\begin{aligned}
&\max_{\mathbf{X}_{\cdot,i},\sigma^2} \mathbb{E}_{q^*}[\frac{-1}{2\sigma^2}(\mathbf{Z}_i - \mathbf{X}_{\cdot,i}\mathbf{W}_{i,\cdot})^\dagger(\mathbf{Z}_i - \mathbf{X}_{\cdot,i}\mathbf{W}_{i,\cdot}) \\
&\qquad -nd\log\sigma^2] \\
\equiv &\max_{\mathbf{X}_{\cdot,i},\sigma^2} \frac{-1}{2\sigma^2}\mathcal{V}(\mathbf{X}_{\cdot,i}) - nd\log\sigma^2, \\
\text{where } &\mathcal{V}(\mathbf{X}_{\cdot,i}) = \mathbf{X}_{\cdot,i}^\dagger \mathbf{X}_{\cdot,i} \mathrm{tr}(\mathbf{c}^i \mathbf{c}^{i\dagger} + \mathbf{D}^i) - 2\widehat{\mathbf{c}^i}^\dagger (\mathbf{X}_{\cdot,i}^\dagger \mathbf{Z}_i).
\end{aligned}$$

Clearly , $\mathbf{X}$ and σ^2 can be updated separately, and in closed form by taking the gradient and setting to 0:

$$\mathbf{X}^*_{\cdot,i} = \frac{\widehat{\mathbf{c}^i}^\dagger \mathbf{Z}_i}{\mathrm{tr}(\mathbf{c}^i \mathbf{c}^{i\dagger} + \mathbf{D}^i)}, \tag{4.11}$$

$$\sigma^{*2} = \frac{\mathcal{V}(\mathbf{X}_{\cdot,i})}{2nd}. \tag{4.12}$$

Algorithm 7 EM Algorithm for SparsePCA

Input: $k, r, \mathbf{C}, \mathbf{T}$
Initialize $\forall j \neq 1, \mathbf{X}_{\cdot,j} = 0$, $\mathbf{X}_{\cdot,1}$ randomly

while not converged **do**
  for $i = 1 \dots r$ **do**
    $\mathbf{Z}_i = \mathbf{T} - \sum_{j \neq i} \mathbf{X}_{\cdot,j} \mathbf{c}^j$
    E-Step
    Init: $\mathsf{K}_i^* = \{\}$
    for $j = 1 \dots k$ **do**
      Update K_i^*:
      $\mathsf{K}_i^* = \arg\max$ Eq. 4.8 over $\mathsf{K}_i^* \bigcup \{t\}$,
        $\forall t \in [d], t \notin \mathsf{K}_i^*$
    end for
    Use Equation 4.9 to update $\mathbf{c}^i$ and $\mathbf{D}^i$ for q_i^*
    M-Step
    Update $\mathbf{X}_{\cdot,i}$ using Equation 4.11
    Update σ^2 using Equation 4.12
  end for
end while
return$(\mathbf{q}^*, \mathbf{X}, \sigma^2)$

Algorithm 7 delineates the entire algorithm stepwise. We have explained the Sparse Probabilistic PCA setup first for simplicity. We now present further applications that arise by modifying the variational E-step to incorporate structured sparse constraints developed in Section 3.3.2.

4.4 Applications: Probabilistic Models with Matroid Constrained Variables

While the class of probabilistic models that admit a representation via matroid constraints is quite broad, we consider special cases in detail (i) group

sparse principal components analysis, (ii) sparse canonical correlation analysis.

4.4.1 Group Sparse Probabilistic Principal Components Analysis

Recall that Probabilistic PCA aims to factorize a matrix $\mathbf{T} \in \mathbb{R}^{n\times}$ as $\mathbf{T} \approx \mathbf{x}\mathbf{w}^\top$, where $\mathbf{x} \in \mathbb{R}^n$ is a deterministic vector, and $\mathbf{w} \in \mathbb{R}^d$ is a random variable. For simplicity, we only consider the rank 1 case i.e. where $\mathbf{x}, \mathbf{w}$ are vectors. The general matrix case follows by a joint estimation procedure within using our framework. The generative model for the observed data matrix is $\mathbf{T} = \mathbf{x}\mathbf{w}^\top + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We consider the case where the prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$, and in addition, $\mathbf{w}$ is assumed be sparse. Let $\theta = \{\mathbf{x}, \sigma\}$ represent the set of deterministic parameters.

We now derive the explicit equations to apply Algorithm 5. The posterior $\mathrm{p}(\mathbf{w}|\mathbf{T};\theta)$ is Gaussian with $\mathrm{p} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}^{-1} = \mathbf{C}^{-1} + \frac{\|\mathbf{x}\|_2^2}{\sigma^2}$, and $\boldsymbol{\mu} = \frac{1}{\sigma^2}\boldsymbol{\Sigma}\mathbf{T}^\top\mathbf{x}$. Define $\mathbf{r} := \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$. Expanding the KL divergence for information projection from (3.9) yields that the support selection requires the following submodular maximization problem:

$$\max_{\left\{\mathsf{S}\subset[r],\, \mathsf{S}=\bigcup_{i\in\mathsf{S}}\mathsf{G}_i,\, |\mathsf{S}|\leq k\right\}} \mathbf{r}_{\mathsf{s}}^\top[\boldsymbol{\Sigma}^{-1}]_{\mathsf{s}}\mathbf{r}_{\mathsf{s}} - \log\det[\boldsymbol{\Sigma}^{-1}]_{\mathsf{s}}.$$

The resulting approximate posterior is given by the respective conditional $\mathrm{q}^*(\mathbf{w}) = \mathrm{p}(\mathbf{w}|\mathbf{w}_{\mathsf{S}^c} = 0)$.

4.4.2 Sparse Probabilistic Collective Matrix Factorization (Sparse PCMF)

Collective Matrix Factorization [Singh and Gordon, 2008, Klami et al., 2013a] is a multiview generalization of PCA. It is typically used to learn joint low rank factorizations with shared entities. The model is closely related to CCA [Witten et al., 2009], and its probabilistic counterpart [Bach and Jordan, 2005, Archambeau and Bach, 2008]. The models are often used interchangeably, though there is a subtle difference. In their probabilistic counterparts CCA assumes full covariance matrix across dimensions while the CMF makes a simpler assumption of an isotropic Gaussian as noise [Klami et al., 2013b]. Both the models are used for studying cross relational effects. We chose to model sparse CMF over sparse CCA to illustrate the application of our framework under another matroidal constraint, namely the partition matroid. We note that sparse probabilistic CCA is within the purview of our framework, albeit it requires a bit more complicated constraint set and algorithm.

We describe the setup for CMF next. Instead of observing single view as an $n \times d$ matrix, or a single *view*, multiple views of the same entities are observed. Hence, we observe n samples of dimensions $d_1, d_2, \ldots, d_v$ as matrices $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_v$ each of which are one of the v views of the observed. The generative model assumes an underlying parameter $\mathbf{x} \in \mathbb{R}^n$ shared among all the views, and the random variables $\{\mathbf{w}_i \in \mathbb{R}^{d_i}, \forall i \in [v]\}$. As in Section 4.4.1, we note that $\mathbf{x}, \{\mathbf{w}_i\}$ can be matrices in general. To emphasize the proposed greedy information projection, we focus on modeling for the top-1

component. The random variables are drawn from Gaussian distribution as $\mathbf{w}_i \sim \mathcal{N}(\underline{0}, \mathbf{C}_i) \forall i \in [v]$, and each of the view is generated as $\mathbf{T}_i = \mathbf{x}\mathbf{w}_i + \epsilon$, where the noise is $\epsilon \sim \mathcal{N}(0, \sigma^2)$ For our experiments, $\mathbf{C}_i$ is set from domain knowledge (see Section 4.5.1.3), and σ^2 allows for additional isotropic variation to capture residuals from the cross correlation. We wish to infer sparse $\mathbf{w}_i$ so that $\forall i \in [v], |\text{supp}(\mathbf{w}_i)| \leq k_i$ for the supplied k_i. The parameters are optimized using an EM algorithm. The variational E-step can be formulated to honor the sparsity constraints on the random variables. We next that show that the variational E-step solves a submodular maximization problem subject to a partition matroidal constraint.

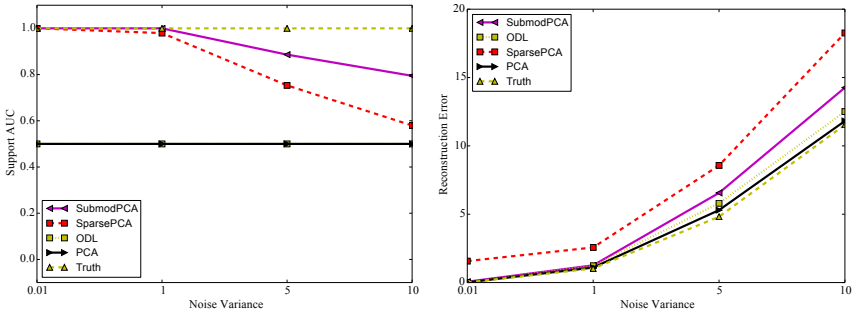
We now map the sparse PCMF problem to the partition matroidal constrained optimization. Let $\mathbf{T} = [\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_v]$ be the matrix of size $n \times (\sum_i d_i)$ constructed by stacking all the observed views column-wise. Similarly, $\mathbf{w} = [\mathbf{w}_1; \mathbf{w}_2; \ldots; \mathbf{w}_v]$ be the vector obtained by end-to-end concatenation of random variable vectors of all views. Define $\mathbf{C} \in \mathbb{R}^{(\sum_i d_i) \times (\sum_i d_i)}$ as the block diagonal matrix with $\mathbf{C}_i$ as its block. The generative model of PCMF can now be equivalently and succinctly encoded as $\mathbf{T} = \mathbf{x}\mathbf{w}^\top + \epsilon$ where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$, and, $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Further, the partition matroid is easy to construct with $\mathsf{N} = [\sum_i d_i]$, and A_i to to be the respective index set of $\mathbf{w}_i$ in $\mathbf{w}$. Again, proceeding as in Section 4.4.1, the submodular maximization problem can be written as:

$$\max_{\{\mathsf{S} \in \mathsf{E},\, \text{Matroid}(\mathsf{N},\mathsf{E})\}} \mathbf{r}_\mathsf{s}^\top [\mathbf{\Sigma}^{-1}]_\mathsf{s} \mathbf{r}_\mathsf{s} - \log\det[\mathbf{\Sigma}^{-1}]_\mathsf{s}.$$

Hence, Algorithm 3 or equivalently Algorithm 4 can be used for sparse infer-

ence. We focus on sparse PCMF for this manuscript. However, it should be easy to the see that further extension to group sparse PCMF is straightforward by modifying the constraining partition matroid appropriately.

4.5 Experiments



(a) Support Recovery as a function of noise variance

(b) Reconstruction error as a function of noise variance

Figure 4.1: Performance on Simulated Data for Sparse PCA. Our method SubmodPCA consistently performs better than established baselines for Support Recovery and Reconstruction error.

For the case of sparse PCA, we would not know the true underlying support for the principal components in the real data sets. Hence, we first validate our model on simulated data sets. We tested the proposed algorithm and some competing baselines on several instances of toy data generated as follows. We fix the number of data points $n = 100$, the ambient dimension size $d = 1000$, the rank $r = 5$, and the sparsity at $k = 20$. We draw r principal components from a Gaussian distribution in the ambient space and zero out all but k randomly chosen dimensions in each of them to form $\mathbf{W}$. We draw

$\mathbf{X}$, the linear weights, independently from $\mathcal{N}(0, 1)$. Finally, for each entry of $\mathbf{T} = \mathbf{XW}$, a noise value is added that is drawn from $\mathcal{N}(0, \sigma^2)$ for various values of σ^2. For metrics, firstly we report the Receiver Operator Characteristic Area Under the Curve (ROC-AUC) on the support recovery of sparse matrix. Secondly, we look at the reconstruction error as the average $||\mathbf{T} - \hat{\mathbf{X}}\hat{\mathbf{W}}||$, where $\hat{\mathbf{X}}$ and $\hat{\mathbf{W}}$ indicate the respective fitted matrices for each of the methods. We compare against Online Dictionary Learning (ODL) [Mairal et al., 2009], and scikit's sparsePCA and standard PCA. We use scikit's implementation for thes [Pedregosa et al., 2011]. 'Truth' in both the graphs is the value obtained using the correct generating parameters. The presented models are the ones that are usually considered for recovering underlying bases, or for sparse reconstruction.

The results are summarized in Figure 4.1. We simulate 10 different data sets and graph the average values of AUC and reconstruction error. The figure shows how methods such as PCA perform well on reconstruction error but are not sparse as they overfit by including the noise dimensions as well. On the other hand, scikit's sparsePCA does reasonably well on capturing the underlying support but does not reconstruct well. Our method (SubmodPCA) does well on reconstruction error while also consistently recovering support, and degrades more gracefully as the noise increases.

4.5.1 Functional Neuroimaging Data

Functional magnetic resonance imaging (fMRI) is an important tool for non-invasive study of brain activity. fMRI studies involve measurements of blood oxygenation (which are sensitive to the amount of local neuronal activity) while the participant is presented with a stimulus or cognitive task. Neuroimaging signals are then analyzed to identify which brain regions which exhibit a systematic response to the stimulation, and thus to infer the functional properties of those brain regions [Poldrack et al., 2011]. Functional neuroimaging data sets typically consist of a relatively small number of correlated high dimensional brain images. Hence, capturing the inherent structural properties of the imaging data is critical for robust inference. These properties vary based on the task at hand, and so many different sparsity constrained problems arise for different tasks. Furthermore, many traditional models do not perform well on the high dimensional fMRI data sets. As such, fMRI data sets are well suited to test and validate our framework.

4.5.1.1 Sparse PCA on Resting State data

Resting state fMRI data are commonly analyzed in order to identify coherently modulated brain networks that reflect intrinsic brain connectivity, which can vary in association with disease and phenotypic variables. We examined the performance of the present method on a resting-state fMRI scan lasting 10 minutes (3T whole-brain multiband EPI, TR=1.16 secs, 2.4 mm resolution), obtained from a healthy adult subject [Poldrack et al., 2015]. Data

were processed using a standard processing stream including motion correction and brain extraction (FSL).

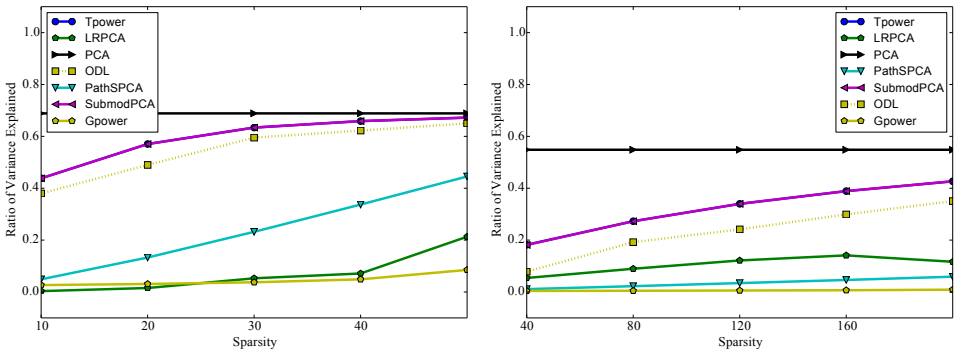
The data originally captured has 518 data points, and over 100,000 dimensions. The ambient set of dimensions are clustered to fewer dimensions using the spatially constrained Ward hierarchical clustering approach of [Michel et al., 2012], to produce three smaller dimensional data sets with 100, 1000, 10000 dimensions. This makes the data set challenging to deal with because we have cases where the dimensionality exceeds the number of datapoints.

We examined the support recovered from these data after estimating four components using our method. The first three components were largely restricted to regions reflecting motion artifacts, which suggests that this method may have utility in the detection and removal of artifacts from fMRI data (similar to previous use of ICA by Tohka et al. [2008]). Figure 4.3 shows the brain map generated using the first principal component extracted using our algorithm. For the three data sets, we compare the ratio of variance explained by the k-sparse first principal component vector (i.e. number of non-zero entries is k) to the total variance in the data set, for varying values of k. We compare against methods: Generalized Power Method [Journée et al., 2010] (Gpower), PCA via Low rank [Papailiopoulos et al., 2013] (LRPCA), Truncated Power Method [Yuan and Zhang, 2013] (Tpower), Online Dictionary Learning [Mairal et al., 2009] (ODL) and Full Regularized Path Sparse PCA [d'Aspremont et al., 2007] (PathSPCA). For comparison, we run the standard PCA (non-sparse), and plot the ratio of explained variance along with all the above mentioned

methods. Figure 4.2 shows the plots for all the three data sets. Note that Gpower and ODL take a regularization parameter rather than sparsity level directly. For both of them, the regularization parameter was adjusted to reach the intended sparsity level and those results are reported. For LRPCA, the authors had implementation for rank =1 and higher ranks. The numbers we report are for rank=2 for d=100 and d=1000 and rank=1 for d=10000. This is because rank=2 for d=10000 was too slow and did not finish after 2 days. We did not notice significant difference in numbers between rank=1 and rank=2 for lower d. The plots clearly show that our method (SubmodPCA) performs consistently at least as well as any of the other sparse methods.

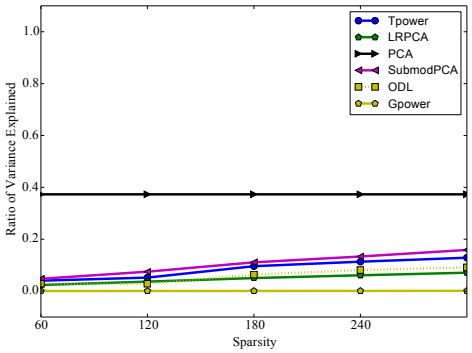
4.5.1.2 Group Sparse PCA on Neurovault data

A key question in functional neuroimaging is the extent to which task brain measurements incorporate distributed regions in the brain. One way to tackle this hypothesis is to decompose a collection of task statistical maps and examine the shared factors. Smith et al. [2009] considered a similar question using the brain map database decomposed via ICA, showing correspondence between task activation factors and resting state factors. Following their approach, we downloaded 1669 fMRI task statistical maps from neurovault [Gorgolewski et al., 2015]. Each image in the collection represents a standardized statistical map of univariate brain voxel activation in response to an experimental manipulation. The statistical maps were downsampled from $2mm^3$



(a) d=100

(b) d=1000



(c) d=10000

Figure 4.2: Performance metrics for various Sparse PCA approaches on fMRI resting state data (Section 4.5.1.1)

voxels to $3mm^3$ voxels using the nilearn python package[1]. We then applied the standard brain mask, removing voxels outsize of the grey matter, resulting in d=65598 variables. We incorporate smoothness via spatial precision matrix $\mathbf{C}^{-1}$ on the prior on $\mathbf{W}$ which is generated by using the adjacency matrix of the three dimensional brain image voxels. This directly corresponds to the observation that nearby voxels tend to have similar functional behavior.

[1] `http://nilearn.github.io`
`https://www.frontiersin.org/articles/10.3389/fninf.2014.00014/full`

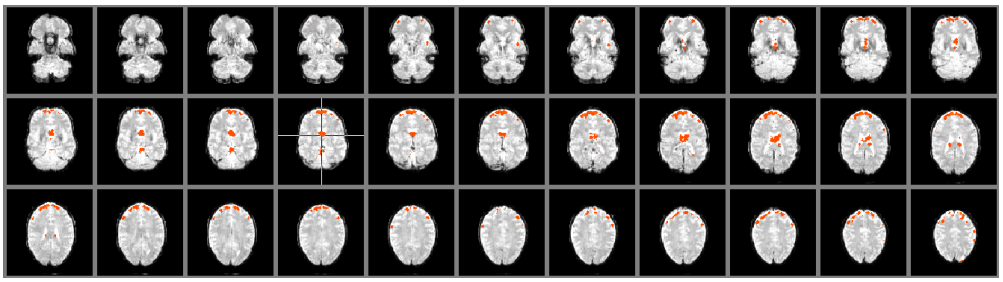
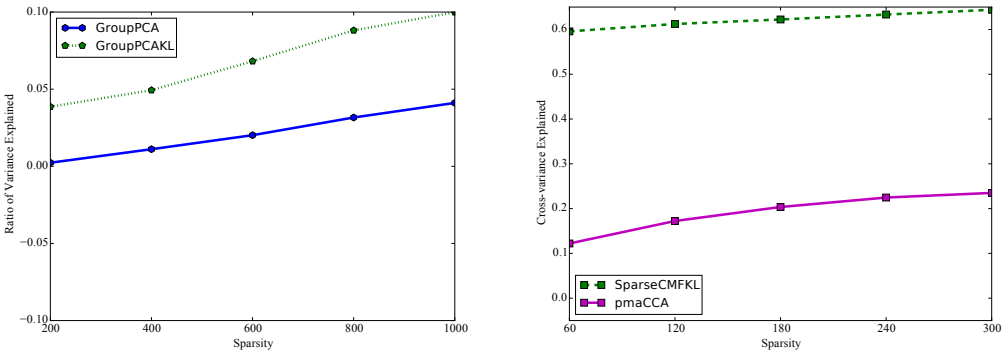


Figure 4.3: A projection of the first sparse component (shown in red) obtained using SubmodPCA (Section 4.5.1.1) onto the mean fMRI image. The component is seen primarily in regions at the frontal surface as well as in the ventricles, consistent with motion artifact.

While our greedy algorithm can easily scale to dimensionality of size 65598, the matlab implementation of the baseline is not as scalable. We cluster the original set of dimensions to d =10000 dimensions using the spatially constrained Ward hierarchical clustering approach of Michel et al. [2012]. We further apply the same hierarchical clustering to group the dimensions into 500 groups, with group sizes ranging from 1 to 1500 with average group size close to 20. We apply our information projection based Group Sparse PCA algorithm (GroupPCAKL). The group sparse constraint specifies that each group can be either wholly included or completely discarded from the model. Our algorithm adheres to this specification. It is possible to have a soft version of the constraint which allows for sparsity within each chosen group. This is typically imposed as a regularization trade-off between sparsity across and within groups. We compare against the Structured Sparse PCA algorithm (GroupPCA) of Jenatton et al. [2010], which is considered state of the art algorithm for group sparse PCA. Like in the case of Sparse PCA, we report the



(a) Group Sparse PCA performance on the Neurovault data

(b) Cross-Correlation on n-back Human Connectome data

Figure 4.4: Performance metrics for Structured Sparse Factor models

ratio of variance explained by the top k-sparse eigenvector at different values of k and show superior performance of GroupPCAKL in Figure 4.4a. The GroupPCAKL provides significant lifts (about times the variance explained) over groupPCA consistently across different sparsity levels.

4.5.1.3 Collective Matrix Factorization on Human Connectome

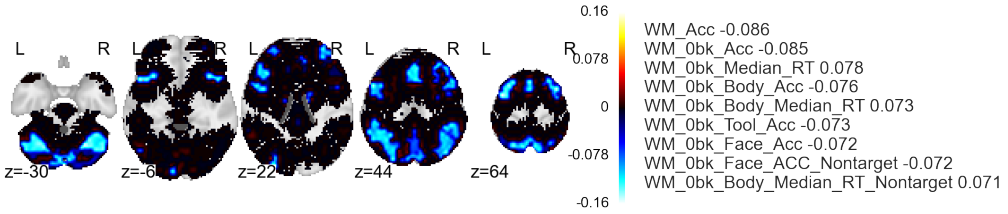
Another interesting question that the neuroscientists are interested to address is about the association of human brain function to human behavior. The brain function and the human behavior can be thought of as two *views* of underlying latent traits. This intuition suggests possible application of the cross correlation based approaches. We make use of the Human Connectome Project data (HCP) [Van Essen et al., 2013] for this purpose. It consists of large number of samples of high quality brain imaging and behavioral information collected from several healthy adults. We specifically use two data sets

of different tasks - 2K (2 Back vs 0 Back contrast, measures working memory), and REL-match (REL vs MATCH contrast, measures relational processing)[2]. We download and extract brain statistical maps (a statistical map is a summary of each voxel in the brain in response to externally applied controlled stimulus) and respective behavioral variances from 497 adult subjects. Each subject has 380 behavioral variables, 27000 downsampled voxels. Further details on the task are available in the HCP documentation [Van Essen et al., 2013]. On the extracted maps, we perform the standard preprocessing for motion correction, and image registration to the MNI template for consistency of comparisons across subjects. The resulting maps we downsampled in the similar way as the Neurosynth data.

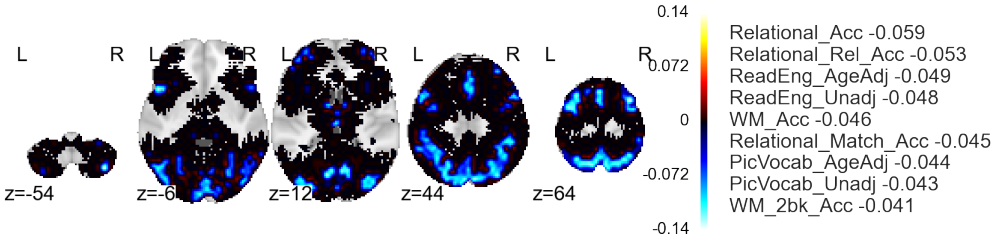
As before, to incorporate smoothness we use the spatial correlation matrix as the prior on the factors of view of statistical map. For the view of behavioral data, we use an identity matrix as the respective prior covariance matrix. We apply our Information Projection based Sparse CMF (SparseCMF-KL) approach and compare it against the Sparse CCA algorithm developed by Witten et al. [2009] (pmdCCA) which is used in its original or slightly modified form as state of the art in many neuroscience and biomedical applications. For quantitative comparison, we use the n-back task data set to report the cross-variance explained which is defined as follows. If $\mathbf{X}, \mathbf{Y}$ are the two views, and $\mathbf{u}, \mathbf{v}$ are the respective (possible sparse) factors, the cross-

[2]https://wiki.humanconnectome.org/display/PublicData/ Task+fMRI+Contrasts

variance is defined as : $\frac{\mathbf{u}^\top \mathbf{X}^\top \mathbf{Y}\mathbf{v}}{|\mathbf{u}^\top \mathbf{X}\mathbf{u}|^{\frac{1}{2}} |\mathbf{v}^\top \mathbf{Y}\mathbf{v}|^{\frac{1}{2}}}$. Note that the normalization ensures that the results are not driven by over estimating the within-view variance. We present strong performance of SparseCMFKL on the metric in Figure 4.4b. For lower sparsity (around 60-sparse) we obtain gains of the order of more than 4 times over pmdCCA. For higher sparsity levels, the order of the gap decreases a bit, but SparseCMFKL maintains a much stronger performance. We also present the extracted voxels and the respective qualitative interpretations on the 2-back and relational task in Figure 4.5a, 4.5b respectively. We note that applying pmdCCA on the same data sets yields inconsistent brainmaps.



(a) The first factor from 2-back task. Neural support is seen in a number of frontal and parietal regions and cerebellum, consistent with cognitive control systems usually engaged by the task. Behavioral correlates including both reaction time and accuracy on the task, showing greater neural engagement associated with slower and less accurate performance.



(b) The first factor from relational reasoning task. Neural support is observed in frontal, parietal, and occipital cortex. Behavioral correlates captured both performance on this particular task, as well as independent measures related to higher cognitive functions including working memory capacity, vocabulary, and reading.

Figure 4.5: Extracted factors using Sparse CMF on the Human Connectome (Section 4.5.1.3)

Chapter 5

Prototype Selection for Interpretation in Machine Learning

It has long been established that using examples to enable interpretability is one of the most effective approaches for human learning and understanding [Newell and Simon, 1972, Cohen et al., 1996, Kim et al., 2014]. The ability to interpret using examples from the data can lead to more informed decision based systems and a better understanding of the inner workings of the model [Koh and Liang, 2017, Kim et al., 2016]. We present two methods for prototype/example selection. Both methods employ density approximation algorithms to select a few examples that closely approximate a known data distribution. The first method uses Bayesian Quadrature and focuses on interpreting model predictions on the test set (Section 5.1). The second method selects examples to approximate the training data distribution using kernel herding (Section 5.2).

However, examples are not enough. Relying only on examples to explain the models' behavior can lead over-generalization and misunderstanding. Examples alone may be sufficient when the distribution of data points are 'clean' – in the sense that there exists a set of prototypical examples which

sufficiently represent the data. However, this is rarely the case in real world data. For instance, fitting models to complex datasets often requires the use of regularization. While the regularization adds bias to the model to improve generalization performance, this same bias may conflict with the distribution of the data. Thus, to maintain interpretability, it is important, along with prototypical examples, to deliver insights signifying the parts of the input space where prototypical examples do not provide good explanations. We call the data points that do not quite fit the model *criticism* samples. Together with prototypes, criticism can help humans build a better mental model of the complex data space.

Bayesian model criticism (BMC) is a framework for evaluating fitted Bayesian models, and was developed to to aid model development and selection by helping to identify where and how a particular model may fail to explain the data. It has quickly developed into an important part of model design, and Bayesian statisticians now view model criticism as an important component in the cycle of model construction, inference and criticism [Gelman et al., 2014]. Lloyd and Ghahramani [2015] recently proposed an exploratory approach for statistical model criticism using the maximum mean discrepancy (MMD) two sample test, and explored the use of the *witness function* to identify the portions of the input space the model most misrepresents the data. Instead of using the MMD to compare two models as in classic two sample testing [Gretton et al., 2012], or to compare the model to input data as in the Bayesian model criticism of Lloyd and Ghahramani [2015], we consider a novel

application of the MMD, and its associated witness function as a principled approach for selecting *prototype* and *criticism* samples.

Our key contributions are as follows:

- We leverage the BMC framework to generate explanations for machine learning methods using the MMD statistic as a measure of similarity between points and potential prototypes
- We provide sufficient conditions for the submodularity to hold for example selection using MMD
- We propose a novel method to select salient training data points that explain test set predictions.
- To solve the resulting combinatorial problem, we develop new approximation guarantees for greedy Sequential Bayesian Quadrature. Our analysis also yields applicability of more scalable algorithm variants for SBQ with provable approximation bounds. These theoretical insights may be of independent interest.
- In addition to prototypes, we select criticism samples i.e. samples that are not well-explained by the prototypes using a regularized witness function score.
- To highlight the practical impact of the our interpretability framework, we present its application to practical tasks.

- We also present results from a human subject pilot study which shows that including the criticism together with prototypes is helpful for an end-task that requires the data-distributions to be well-explained.

5.1 Prototype Selection using Fisher Kernels

In this section, we present our method to select sample representatives using Fisher kernels. For a loss function $\ell(\theta, \mathbf{x})$, where θ are the parameters of the model and $\mathbf{x}$ is the data, to train a parametric model one would minimize the expected loss:

$$\min \mathbb{E}_{p(\mathbf{x})} \ell(\theta, \mathbf{x}), \tag{5.1}$$

where $p(\mathbf{x})$ is the data distribution. Since we usually do not have access to the true data distribution, $p(\mathbf{x})$ is typically the empirical data distribution $p(\mathbf{x}) = \frac{1}{n}\delta(\mathbf{x})$, where $\delta(\cdot)$ is 1 if $\mathbf{x}$ exists in the dataset, and 0 otherwise, and n is the size of the dataset. Our goal in this work is to approximate the integral (5.1) over the test or validation set (which specifies the distribution p for us) using a weighted sum of a *few* points from the *training* dataset (5.2) using an algorithm called *Sequential Bayesian Quadrature*.

For the kernel function in the GP prior in Bayesian Quadrature, we use the Fisher kernel of the trained parametric model. SBQ selection strategy inherently establishes a trade off between selecting data points that are representative of the parametric fit and diversity of the selected points. To see this, consider the SBQ cost function (5.3). At every new selection $\mathbf{x}_{j+1}$, the

cost function pushes to select data points which are clustered closer together in the feature mapping space to increase the value of $\mathbf{z}$ which in turn decreases variance. However, selecting points close to each other decreases the eigenvalues of $\mathbf{K}^{-1}$ thereby increasing variance [Huszar and Duvenaud, 2012]. We briefly describe Bayesian Quadrature and Fisher Kernels before delineating the algorithm.

5.1.1 Fisher Kernels

The notion of similarity that Fisher kernels employ is that two similar objects would have similar gradients in the parameters of the model. If two objects are *structurally* similar, then slight perturbations in the neighborhood of the fitted parameters $\hat{\theta} := \arg\max \log p(\mathbf{X}|\theta)$, would impact the fit of the two objects similarly. In other words, with $\mathbf{f}_i := \frac{\partial \log p(\mathbf{X}_i|\theta)}{\partial \theta}|_{\theta=\hat{\theta}}$, for an object $\mathbf{X}_i \to \mathbf{f}_i$ can be interpreted as a *feature mapping* which can then be used to define a similarity kernel by a weighted dot product:

$$\kappa(\mathbf{X}_i, \mathbf{X}_j) := \mathbf{f}_i^\top \mathcal{I}^{-1} \mathbf{f}_j,$$

where the matrix $\mathcal{I} := \mathbb{E}_{p(\mathbf{X})}[\frac{\partial \log p(\mathbf{X}|\theta)}{\partial \theta}^\top \frac{\partial \log p(\mathbf{X}|\theta)}{\partial \theta}]$ is the Fisher information matrix. The information matrix serves to re-scale the dot product, and is often taken as identity as it loses significance in limit [Jaakkola and Haussler, 1999]. The corresponding kernel is then called the *practical* Fisher kernel and is often used in practice. We note, however, that dropping $\mathcal{I}$ had significant impact on performance in our method, so we employ the full kernel

rather than the practical kernel. Another interpretation of the Fisher kernel is that it defines the inner product of the directions of gradient ascent over the Riemannian manifold that the generative model lies in [Shawe-Taylor and Cristianini, 2004].

While appropriate feature mapping is crucial for predictive tasks, we observe that it is also is vital for interpretability. We use Fisher Kernels as the embedding space before we select prototypes. Fisher kernels are ideal for this task because they seamlessly integrate supervision from the trained model that we wish to interpret for into the process of mapping features to a more amenable space. To further motivate that such a task can not be trivially performed by a something like a parameter sweep over RBF kernels that does not include supervision, we perform a simple toy experiment illustrated in Figure 5.1.

5.1.2 Bayesian Quadrature

Bayesian quadrature [O'Hagan, 1991] is a method used to approximate the expectation of a function by a weighted sum of a few evaluations of the said function. Say a function $f : \mathcal{X} \to \mathbb{R}$ is defined on a measurable space $\mathcal{X} \subset \mathbb{R}^d$. Consider the integral:

$$\mathbb{E}[f(\mathbf{x})] = \int_{\mathcal{X}} f(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \sum_{i=1}^{n} w_i f(\mathbf{x}_i), \tag{5.2}$$

where w_i are the weights associated with function evaluations at $\mathbf{x}_i$. Using $w_i = 1/n$ and randomly sampling $\mathbf{x}_i$ recovers the standard Monte Carlo

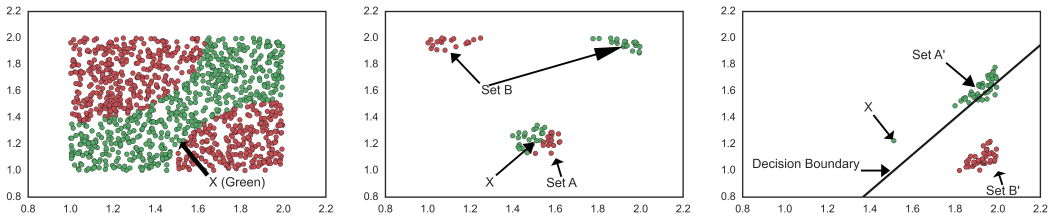


Figure 5.1: A toy experiment to illustrate the usefulness of Fisher space mapping. [Left] 1200 samples on U[1,2] $\times$ U[1,2] with two labels - Green and Red as illustrated. A specific green point X is selected for further experiment. [Mid] Closest 40 (Set A) and farthest 40 points (Set B) in terms of RBF kernel similarity. A distance based kernel such as RBF would yield these points as most and least similar to X respectively. [Right] Closest 40 (Set A') and farthest 40 (Set B') to X in terms of the Fisher kernel similarity. The decision boundary for fitted logistic regression is also presented. It predicts everything below it as red, and everything above it as green. The Fisher "closeness" here takes into account the label of the points as well as the log-likelihood gradient on the contour of the loss function and its direction for each point. Note that the gradient and Fisher similarity with all other points will be 0 for points exactly on the boundary.

integration. Other methods include kernel herding [Chen et al., 2010] and quasi-Monte carlo [Dick and Pillichshammer, 2010], both of which use $w_i = 1/n$ but use specific schemes to draw $\mathbf{x}_i$. Bayesian quadrature allows one to consider a non-uniform w_i given a functional prior for $f(\cdot)$. The samples $\mathbf{x}_i$ can then be chosen as the ones that minimize the posterior variance [Huszar and Duvenaud, 2012]. The corresponding weights can be calculated directly from the posterior mean. We impose a Gaussian Process prior on the function as $f \sim \mathrm{GP}(0, k)$ with a kernel function $k(\cdot, \cdot)$. The algorithm SBQ proceeds as follows. Say we have already chosen n points: $\mathbf{x}_i, i \in [n]$. The posterior of f given the evaluations $f(\mathbf{x}_i)$ is a Gaussian with the mean function:

$$\hat{f}(\mathbf{x}) = \mathbf{k}^\top \mathbf{K}^{-1} \mathbf{f},$$

where $\mathbf{f}$ is the vector of function evaluations $f(\mathbf{x}_i)$, $\mathbf{k}$ is the vector of kernel evaluations $k(\mathbf{x}, \mathbf{x}_i)$, and $\mathbf{K}$ is the kernel matrix with $\mathbf{K}_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$. Note that the weights in (5.2) can be written as $w_i = \sum_j \mathbf{k}_j [\mathbf{K}^{-1}]_{ij}$.

Now that we know how to calculate the corresponding weights of the selected points, we now focus on how to select these points $\mathbf{x}_i$. The quadrature estimate provides not only the mean, but the full distribution as its posterior. The posterior variance can be written as:

$$\mathrm{cov}(\mathbf{x}, \mathbf{y}) = k(\mathbf{x}, \mathbf{y}) - k(\mathbf{x}, \mathbf{X}) \mathbf{K}^{-1} k(\mathbf{X}, \mathbf{y}),$$

where $\mathbf{X}$ is the matrix formed by stacking $\mathbf{x}_i$, and the kernel function

notation is overloaded so that $k(\mathbf{X}, \mathbf{y})$ represents the column vector obtained by stacking $k(\mathbf{x}_i, \mathbf{y})$. The posterior over the function f also yields a posterior over the expectation over f defined in (5.2). For convenience, define the set $\mathsf{S}_j := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_j\}$. Say $Z(\mathsf{S}_j) := \sum_j w_j f(\mathbf{x}_j)$. Then, it is straightforward to see $\mathbb{E}[Z(\mathsf{S}_n)] = \mathbf{z}^\top \mathbf{K}^{-1} f(\mathbf{X})$, where $\mathbf{z}_i := \int k(\mathbf{x}, \mathbf{x}_i) p(\mathbf{x}) d\mathbf{x}$. We can write the variance of $Z(\mathsf{S}_n)$ as:

$$\mathrm{var}(Z(\mathsf{S}_n)) = \iint k(\mathbf{x}, \mathbf{y}) p(\mathbf{x}) p(\mathbf{y}) d\mathbf{x} d\mathbf{y} - \mathbf{z}^\top \mathbf{K}^{-1} \mathbf{z}. \tag{5.3}$$

The algorithm Sequential Bayesian Quadrature (SBQ) samples for the points $\mathbf{x}_i$ in a greedy fashion with the goal of minimizing the posterior variance of the computed approximate integral:

$$\mathbf{x}_{n+1} \leftarrow \arg\min_{\mathbf{x} \in \mathcal{X}} \mathrm{var}(Z(\mathsf{S}_n \cup \{\mathbf{x}\})).$$

5.1.3 An Efficient Greedy Algorithm

In this section, we provide a practical greedy algorithm to select representative prototypes using SBQ to optimize (5.3). Note that the first term is constant w.r.t to S_n. Moreover, $p(\mathbf{x}) = \frac{1}{n}\delta(\mathbf{x})$. Thus, we can re-write $\mathbf{z}_i = \frac{1}{n} \sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{x}_j)$ for each i in training and each j in the test set. This can be pre-computed by a row or column sum over the kernel of the entire dataset in $O(nm)$ time and stored as vector of size m to speed up later computation, where m is the size of the training set and n is the size of the test set. Our greedy cost function at step $j + 1$ is thus:

$$i^{\star}_{j+1} \leftarrow \underset{\substack{i \in [m] \setminus \mathsf{S}_j \\ \mathsf{S} = \mathsf{S}_j \cup i}}{\operatorname{argmax}} \mathbf{z}_{\mathsf{S}}^{\top} [\mathbf{K}_{\mathsf{SS}}^{-1}] \mathbf{z}_{\mathsf{S}}. \tag{5.4}$$

The solution set is then updated as $\mathsf{S}_{j+1} = \mathsf{S}_j \cup \{i^{\star}_{j+1}\}$. The optimization (5.4) requires an inverse of the kernel matrix of already selected data points which can be painful. However, we can use a result from linear algebra about block matrix inverses to speed up operations.

Proposition 5.1.1. *For an invertible matrix* $\mathbf{A}$*, a column vector* $\mathbf{b}$*, and a scalar* c*, let* $d = c - \mathbf{b}^{\top}\mathbf{A}^{-1}\mathbf{b}$*, then*

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^{\top} & c \end{bmatrix}^{-1} = \frac{1}{d} \begin{bmatrix} d\mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{b}\mathbf{b}^{\top}\mathbf{A}^{-1} & \mathbf{A}^{-1}\mathbf{b} \\ \mathbf{b}^{\top}\mathbf{A}^{-1} & 1 \end{bmatrix}$$

Proposition 5.1.1 allows us to build the inverse of the kernel $\mathbf{K}$ in (5.4) greedily. The full algorithm is presented in Algorithm 8.

Algorithm 8 Greedy Prototype Selection

INPUT: Data $\{\mathbf{x}_i\}$, kernel function $k(\cdot, \cdot)$, number of selections to make k
//Pre-compute $\mathbf{z}_i$
$\mathbf{z}_i = \frac{1}{n} \sum_j k(\mathbf{x}_i, \mathbf{x}_j) \forall i \in$ training and $j \in$ test
// Build solution set S greedily. Maintain current inverse(K) at each iteration as inv$\mathbf{K}$
$\mathsf{S} = \emptyset$, inv$\mathbf{K} = []$
for $i = 1 \ldots k$ **do**
    $j^{\star} = \arg\max_{j \in [m] \setminus \mathsf{S}} \mathbf{z}_j^{\top}$ inv$\mathbf{K}\, \mathbf{z}_j$
    Write $\mathbf{b} = k(\mathbf{X}_{\mathsf{S}}, \mathbf{x}_{j^{\star}})$, $c = k(\mathbf{x}_{j^{\star}}, \mathbf{x}_{j^{\star}})$, $\mathbf{A}^{-1} =$ inv$\mathbf{K}$
    Update: inv$\mathbf{K}$ using Prop. 5.1.1, $\mathsf{S} = \mathsf{S} \cup j^{\star}$
end for
return S

Algorithm 8 obviates the need for taking explicit inverses and only requires an oracle access to the kernel function. Building the vector $\mathbf{z}$ is $O(nm)$. For each $i \in [k]$, finding $j^\star$ requires an argmax over the remaining candidate vectors, and hence is $O(k^2 m)$ (as opposed to $O(k^3 n)$ when taking explicit inverses), and updating inv$\mathbf{K}$ is $O(km)$. The time taken to build the kernel matrix is $O(m^2)$. Thus the total run time is $O(m^2 + k^2 m + mn)$. The algorithm itself is inherently embarrassingly parallelizable over multiple cores.

5.1.4 Analysis

To provide guarantees for Algorithm 9, we show that the normalized set optimization function is $\frac{m}{M}$-weak submodular, where m, M depend on the spectrum of the kernel matrix. Recall our set optimization function (from (5.4)) is:

$$g(\mathsf{S}) := \max_{\mathsf{S} \subset [n] | \mathsf{S}| \leq r} \mathbf{z}_{\mathsf{S}} [\mathbf{K}_{\mathsf{SS}}^{-1}] \mathbf{z}_{\mathsf{S}} \tag{5.5}$$

Note that $g(\emptyset) = 0$, so $g(\cdot)$ is normalized. We shall make use of sparse eigenvalues for a matrix. For a matrix $\mathbf{A}$, the smallest (largest) k-sparse eigenvalues as min (max) of $\frac{\mathbf{x}^\top \mathbf{A} \mathbf{x}}{\mathbf{x}^\top \mathbf{x}}$ such that $\|\mathbf{x}\|_0 \leq k$, and $\mathbf{x} \neq 0$. We present our approximation guarantee next.

Theorem 5.1.2. *Let m and M be the smallest and largest k-sparse eigenvalues of the kernel matrix $\mathbf{K}$ of the training set. If S_G of size k is the set returned*

by Algorithm 8 and $\mathsf{S}^\star$ *of size* r *is the optimal solution of* (5.5)*, then,*

$$g(\mathsf{S}_G) \geq \left(1 - \exp(-\frac{mk}{Mr})\right) g(\mathsf{S}^\star).$$

Proof. We make use of the following lemma that establishes a connection between MMD and Bayesian Quadrature.

Lemma 5.1.3. *[Huszar and Duvenaud, 2012] Let* q *be the distribution established by weights* w_i *of the Bayesian Quadrature over the selected points. Then, the expected variance of the weighted sum in Bayesian Quadration* (5.3) *is equal to* $MMD^2(p, q)$*.*

We can make this explicit in our notation. If $\mathcal{F}$ is an RKHS, we can obtain $\phi(\cdot)$ in closed form, and write the MMD cost function using only the kernel function $K(\cdot, \cdot)$ associated with the RKHS Gretton et al. [2012] as:-

$$\begin{aligned}
\mathrm{MMD}_{\mathcal{F}}(p, q)^2 &= \int_{\mathbf{x}\sim p} \int_{\mathbf{y}\sim p} K(\mathbf{x}, \mathbf{y}) p(\mathbf{x}) p(\mathbf{y}) d\mathbf{x} d\mathbf{y} \\
&\quad - 2 * \int_{\mathbf{x}\sim p} \int_{\mathbf{y}\sim q} K(\mathbf{x}, \mathbf{y}) p(\mathbf{x}) q(\mathbf{y}) d\mathbf{x}\mathbf{y} \\
&\quad + \int_{\mathbf{x}\sim q} \int_{\mathbf{y}\sim q} K(\mathbf{x}, \mathbf{y}) q(\mathbf{x}) q(\mathbf{y}) d\mathbf{x}\mathbf{y} \\
&= \|\mu_p - \sum_i w_i \mathbf{z}_i\|_{\mathcal{F}}^2,
\end{aligned}$$

where, $\mathbf{z}_i := \int k(\mathbf{x}, \mathbf{x}_i) p(\mathbf{x}) d\mathbf{x}$, i ranges over the selected points that define our discrete distribution q. Recall that Bayesian Quadrature deviates from simple kernel herding by allowing for and optimizing over non-uniform

weights w_i.This implies that the weight optimization performs an orthogonal projection of μ_p onto the span of selected points to get μ_q. In other words, the weight optimization is a simple linear regression in the mapped space $\mathcal{F}$, and SBQ is equivalent to a greedy forward selection algorithm in $\mathcal{F}$. To get the final result, we make use of the following recent result from the discrete optimization literature.

Lemma 5.1.4. *[Elenberg et al., 2018, Das and Kempe, 2011] The linear regression function is $\frac{m}{M}$-weak submodular where m and M are min and max k-sparse eigenvalues of the inner product matrix of the features respectively.*

The result of Theorem 5.1.2 implies directly from approximation guarantees of weak submodular functions. □

Theorem 5.1.2 guarantees $(1 - e^{-\frac{m}{M}})$ approximation when $k = r$. In addition, the result also implies that to achieve an ε-approximation to the best r-sized solution, we only need to run Algorithm 8 for $k = O(\log \frac{1}{\varepsilon})$ iterations. Furthermore, the connection to weak submodularity also allows for faster and more scalable algorithms with some compromise on the approximation guarantees e.g. a distribute-and-aggregate greedy algorithm, and a stochastic greedy algorithm [Khanna et al., 2017]. To the best of our knowledge, these variants have not been considered for SBQ before.

5.2 `MMD-critic` for Prototype Selection

Given n samples from a statistical model $\mathsf{X} = \{x_i,\, i \in [n]\}$, let $\mathsf{S} \subseteq [n]$ represent a subset of the indices, so that $\mathsf{X}_\mathsf{S} = \{x_i\ \forall i \in \mathsf{S}\}$. Given a RKHS with the kernel function $k(\cdot, \cdot)$, we can measure the maximum mean discrepancy between the samples and any selected subset using $\mathrm{MMD}^2(\mathcal{F}, X, X_\mathsf{S})$. `MMD-critic` selects prototype indices S which minimize $\mathrm{MMD}^2(\mathcal{F}, X, X_\mathsf{S})$. For our purposes, it will be convenient to pose the problem as a *normalized* discrete maximization. To this end, consider the following cost function, given by the negation of $\mathrm{MMD}^2(\mathcal{F}, X, X_\mathsf{S})$ with an additive bias:

$$\begin{aligned} J_b(\mathsf{S}) =& \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j) - \mathrm{MMD}^2(\mathcal{F}, X, X_\mathsf{S}) \\ =& \frac{2}{n|\mathsf{S}|}\sum_{i\in[n], j\in\mathsf{S}} k(x_i, y_j) - \frac{1}{|\mathsf{S}|^2}\sum_{i,j\in\mathsf{S}} k(y_i, x_j). \end{aligned} \tag{5.6}$$

Note that the additive bias $\mathrm{MMD}^2(\mathcal{F}, X, \emptyset) = \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j)$ is a constant with respect to S. Further, $J_b(\mathsf{S})$ is normalized, since, when evaluated on the empty set, we have that:

$$J_b(\emptyset) = \min_{\mathsf{S}\in 2^{[n]}} J_b(\mathsf{S}) = \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j) - \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j) = 0.$$

`MMD-critic` selects m_* prototypes as the subset of indices $\mathsf{S} \subseteq [n]$ which optimize:

$$\max_{\mathsf{S}\in 2^{[n]}, |\mathsf{S}|\leq m_*} J_b(\mathsf{S}). \tag{5.7}$$

5.3 Model Criticism

In addition to selecting prototype samples, `MMD-critic` characterizes the data points not well explained by the prototypes – which we call the model *criticism*. These data points are selected as the largest values of the witness function (2.7) i.e. where the similarity between the dataset and the prototypes deviate the most. Consider the cost function:

$$L(\mathsf{C}) = \sum_{l \in \mathsf{C}} \left| \frac{1}{n} \sum_{i \in [n]} k(x_i, x_l) - \frac{1}{m} \sum_{j \in \mathsf{S}} k(x_j, x_l) \right|. \tag{5.8}$$

The absolute value ensures that we measure both positive deviations $f(x) > 0$ where the prototypes *underfit* the density of the samples, and negative deviations $f(x) < 0$, where the prototypes *overfit* the density of the samples. Thus, we focus primarily on the magnitude of deviation, rather than its sign. The following theorem shows that (5.8) is a linear function of C.

Theorem 5.3.1. *The criticism function $L(\mathsf{C})$ is a linear function of C.*

Proof. A discrete function is linear if it can be written in the form $F(\mathsf{C}) = \sum_{i \in [n]} w_i \mathsf{1}_{[i \in \mathsf{C}]}$. Consider (5.8) and observe that:

$$\begin{aligned} L(\mathsf{C}) &= \sum_{l \in \mathsf{C}} \left| \frac{1}{n} \sum_{i \in [n]} k(x_i, x_l) - \frac{1}{m} \sum_{j \in \mathsf{S}} k(x_j, x_l) \right| \\ &= \sum_{l \in [n]} \left(\left| \frac{1}{n} \sum_{i \in [n]} k(x_i, x_l) - \frac{1}{m} \sum_{j \in \mathsf{S}} k(x_j, x_l) \right| \right) \mathsf{1}_{[l \in \mathsf{C}]} \\ &= \sum_{l \in [n]} w_l \mathsf{1}_{[l \in \mathsf{C}]}, \end{aligned}$$

where:

$$w_l = \left| \frac{1}{n} \sum_{i \in [n]} k(x_i, x_l) - \frac{1}{m} \sum_{j \in \mathsf{S}} k(x_j, x_l) \right|.$$

□

We found that the addition of a regularizer which encourages a diverse selection of criticism points improved performance. Let $r : 2^{[n]} \mapsto \mathbb{R}$ represent a regularization function. We select the criticism points as the maximizers of this cost function:

$$\max_{\mathsf{C} \subseteq [n] \setminus \mathsf{S}, |\mathsf{C}| \leq c_*} L(\mathsf{C}) + r(\mathbf{K}, \mathsf{C}) \tag{5.9}$$

Where $[n] \setminus \mathcal{S}$ denote all indexes which not include the prototypes, and c_* is the number of criticism points desired. Fortunately, due to the linearity of (2.7), the optimization function (5.9) is submodular when the regularization function is submodular. We encourage the use of regularizers which incorporate diversity into the criticism selection. We found the best qualitative performance using the log-determinant regularizer [Krause et al., 2008]. Let $\mathbf{K}_{\mathsf{C},\mathsf{C}}$ be the sub-matrix of $\mathbf{K}$ corresponding to the pair of indexes in $\mathsf{C} \times \mathsf{C}$, then the log-determinant regularizer is given by:

$$r(\mathbf{K}, \mathsf{C}) = \log \det \mathbf{K}_{\mathsf{C},\mathsf{C}} \tag{5.10}$$

which is known to be submodular. Further, several researchers have found, both in theory and practice [Sharma et al., 2015], that greedy optimization is an effective strategy for optimization. We apply the greedy algorithm for criticism selection with the function $F(\mathsf{C}) = L(\mathsf{C}) + r(\mathbf{K}, \mathsf{C})$.

5.4 Experiments

We present results for the proposed technique `MMD-critic` using USPS hand written digits [Hull, 1994] and Imagenet [Deng et al., 2009] datasets. We quantitatively evaluate the prototypes in terms of predictive quality as compared to related baselines on USPS hand written digits dataset. We also present preliminary results from a human subject pilot study. Our results suggest that the model criticism – which is unique to the proposed `MMD-critic` is especially useful to facilitate human understanding. For all datasets, we employed the radial basis function (RBF) kernel with entries $k_{i,j} = k(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|)$.

The Nearest Prototype Classifier: While our primary interest is in interpretable prototype selection and criticism, prototypes may also be useful for speeding up memory-based machine learning techniques such as the nearest neighbor classifier by restricting the neighbor search to the prototypes, sometimes known as the *nearest prototype classifier* [Bien and Tibshirani, 2011, Kuncheva and Bezdek, 1998]. This classification provides an objective (although indirect) evaluation of the quality of the selected prototypes, and is useful for setting hyperparameters. We employ a 1 nearest neighbor classifier using the Hilbert space distance induced by the kernels. Let $y_i \in [k]$ denote the label associated with each prototype $i \in \mathsf{S}$, for k classes. As we employ normalized kernels (where the diagonal is 1), it is sufficient to measure the pairwise kernel similarity. Thus, for a test point $\hat{x}$, the nearest neighbor

classifier reduces to:

$$\hat{y} = y_{i*}, \text{ where } i^* = \arg\min i \in \mathsf{S} \, \|\hat{x} - x_i\|^2_{\mathcal{H}_K} = \arg\max i \in \mathsf{S} \, k(\hat{x}, x_i).$$

5.4.1 `MMD-critic` evaluated on USPS Digits Dataset

The USPS hand written digits dataset Hull [1994] consists of $n = 7291$ training (and 2007 test) greyscale images of 10 handwritten digits from 0 to 9. We consider two kinds of RBF kernels (i) *global:* where the pairwise kernel is computed between all data points, and (ii) *local:* given by $\exp(-\gamma\|x_i - x_j\|)\mathbf{1}_{[y_i=y_j]}$, i.e. points in different classes are assigned a similarity score of zero. The local approach has the effect of pushing points in different classes further apart. The kernel hyperparameter γ was chosen based to maximize the average cross-validated classification performance, then fixed for all other experiments.

Classification: We evaluated nearest prototype classifiers using `MMD-critic`, and compared to baselines (and reported performance) from Bien and Tibshirani [2011] (abbreviated as PS) and their implementation of K-medoids. Figure 5.2(left) compares `MMD-critic` with global and local kernels, to the baselines for different numbers of selected prototypes $m = |\mathsf{S}|$. Our results show comparable (or improved) performance as compared to other models. In particular, we observe that the global kernels out-perform the local kernels[1]

[1]Note that the local kernel trivially achieves perfect accuracy. Thus, in order to measure generalization performance, we do not use class labels for local kernel *test* instances i.e. we use the global kernel instead of local kernel for *test* instances – regardless of training.

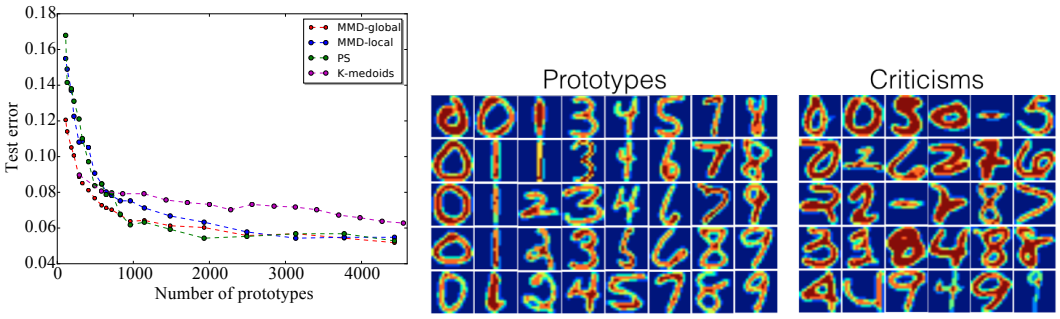


Figure 5.2: Classification error vs. number of prototypes $m = |S|$. `MMD-critic` shows comparable (or improved) performance as compared to other models (left). Random subset of prototypes and criticism from the USPS dataset (right).

by a small margin. We note that MMD is particularly effective at selecting the first few prototypes (i.e. speed of error reduction as number of prototypes increases) suggesting its utility for rapidly summarising the dataset.

Selected Prototypes and Criticism: Fig. 5.2 (right) presents a randomly selected subset of the prototypes and criticism from the `MMD-critic` using the local kernel. We observe that the prototypes capture many of the common ways of writing digits, while the criticism clearly capture outliers.

5.4.2 Data Cleaning: removing malicious training data points

In this section, we present experiments on the MNIST dataset to illustrate the effectiveness of our method in interpreting model behavior for the test population. Some of the handwritten digits in MNIST are hard even for a human to classify correctly. Such points can adversely affect the training of the classifier, leading to lower predictive accuracy. Our goal in this experiment

is to try to identify some such misleading training data points, and remove them to see if it improves predictive accuracy. To illustrate the flexibility of our approach, we focus only on the digits 4 and 9 in the test data which were misclassified by our model, and then select the training data points responsible for those misclassifications.

The MNIST data set [LeCun et al., 1998] consists of images of handwritten digits and their respective labels. Each image is a 28×28 pixel array. There are 70000 images in total, split into 60000 training examples and 10000 test examples. The 10 digits are about evenly represented in both the training and the test data.

For the classification task, we use tensorflow [Abadi et al., 2016] to build a 2 layer convolutional network with 2×2 max pooling followed by a fully connected layer and the softmax layer. The convolutions use a stride of 1 followed by padding of zeros to match the input size. We use dropout to avoid overfitting. The network was trained using the inbuilt AdamOptimizer for 20000 steps of batch size 100 each. For the entire test set, we obtained an accuracy of 0.9922, while for the subset of the test set consisting only of the chosen two digits 4 and 9, the accuracy was 0.9889.

After the training is completed, we obtain the gradients of the training and test data points w.r.t the parameters of the network by passing each point through the trained (and subsequently frozen) network. The obtained gradient vectors are used to calculate the Fisher kernel as detailed in Section 5.1.1. We then employ Algorithm 9 using the newly built Fisher kernel matrix between

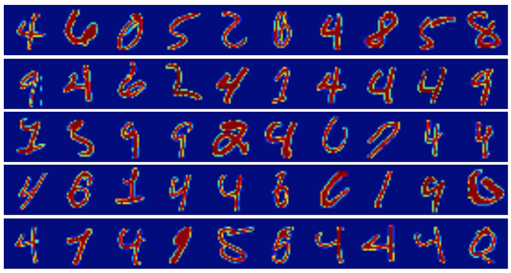
training and test datasets to obtain the top 300 *prototypes* i.e. data points from the training set that our algorithm deems most responsible for misclassifying 4s and 9s.

To check if these points are indeed misguiding the model, we remove the top $50, 100, 200, 300$ of the selected points from the training data and retrain the model to retest on the test set. These numbers are reported as Sel50, Sel100, Sel200, Sel300 in Figure 5.3b. Indeed we see an improvement in the test accuracy till Sel200 indicating the importance of removing the selected potential malicious points from the training set, and a subsequent decay in performance for Sel300 most likely due to removal of too many useful points as well in addition to malicious ones. To compare, we also remove the respective number of points randomly and repeat the experiment. Removal of random points from the training data led to a general decay in the predictive accuracy.

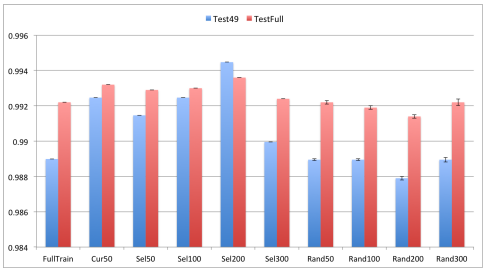
Finally, we manually selected 50 points from the chosen 300 points as the curated set based on how ill-formed the digits were (see Figure 5.3a). Removing these points from the training set before re-training and testing gives predictive accuracy is reported as Cur50 comparable to Sel100, but still worse than Sel200, indicating that the algorithm identified more malicious points in top-200 selected than our manually chosen 50 points.

5.4.3 Fixing Mislabeled Examples

In this experiment, we use our framework to detect and fix mislabeled examples. Labor intenstive labeling tasks naturally result in mislabels, espe-



(a) A subset of selected prototypes responsible for misclassifying 4s and 9s in the test set



(b) Accuracy fractions on test data 4s and 9s (Test49), and the full test set after removing random (Rand), algorithm selected (Sel), or Curated (Cur) prototypes.

Figure 5.3: MNIST experiment for selecting malicious training data points.

cially in real-world datasets. These data points may cause poor performance and degradation of the model at best. We show that our method can be successfully used for this purpose, showing improvement over the recent results shown by Koh and Liang [2017].

We use a small set of correctly labeled validation set to identify examples from the large training set that are likely mislabeled. We first train a classifier on the noisy training set, and predict on the validation set. We then employ Algorithm 9 to identify training examples that were responsible for making incorrect predictions on the validation set. The potentially mislabeled data points are then chosen by the output of our method. Curation is then simulated on the selected examples in order of selections made (similar to by Koh and Liang [2017]), and if the label was indeed wrong, it is fixed. We report on the number of training data points selected vs fixed (the precision metric for incorrectly labeled points) and the respective improvement in

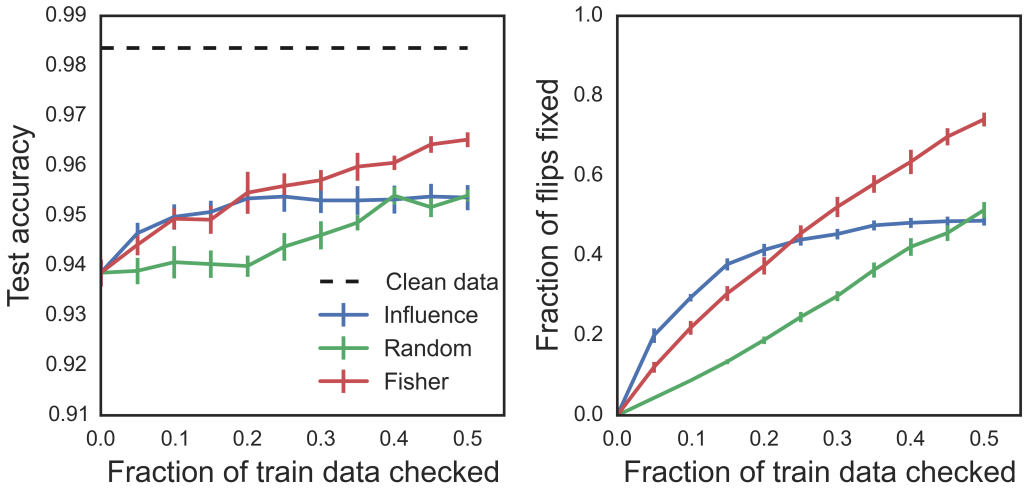


Figure 5.4: Comparison of SBQ compared to Influence functions on the task of fixing flipped labels.

unseen test data accuracy.

For evaluation, we use `enron1` email spam dataset used by Koh and Liang [2017] and compare our results to their reported results. The dataset consists of 4137 training points and 1035 test points. We randomly select 500 data points from the training set as the clean *curated* data. From the remaining training data points, we randomly flip the labels of 20% of the data. We then use our method and the baselines to select several different number of candidates for curation. We report the number of fixes made after these selections and the corresponding test predictive accuracy. The baselines are selection by top self influence measures [Koh and Liang, 2017], and random selection of datapoints. The curation data is used as part of the training by all the methods. No method had access to the test data. As showing in Figure 5.4,

our algorithm consistently performs better in test accuracy and the fraction of flips fixed as more and more data is curated.

5.4.4 Data Summarization

In this section, we perform the task of training data summarization. Our goal is to select a few data samples that represent the data distribution *sufficiently* well, so that a model built on the selected subsample of the training data does not degrade too much in performance on the unseen test data. This task is complimentary to the task of interpretation, wherein one is interested in selecting training samples that explain some particular predictions on the test set. Since we are interested in approximating the test distribution using a few samples from a training set with the goal of predictive accuracy under a given model, our framework of Sequential Bayesian Quadrature using Fisher kernels is directly applicable.

Another method that also aims to do training data summarization is that of coreset selection Huggins et al. [2016], albeit with a different goal of reducing the training data size for optimization speedup while still maintaining guaranteed approximation to the training likelihood. Since the goal itself is optimization speedup, coreset selection algorithms typically employ fast methods while still trying to capture the data distribution by proxy of the training likelihood. Moreover, the coreset selection algorithm is usually closely tied with the respective model as opposed to being a model-agnostic method like ours.

To illustrate that coreset selection falls short on the goal of competitively estimating the data distribution, we employ our framework to the problem of training data summarization under logistic regression, as considered by Huggins et al. [2016] using coreset construction. We experiment using two datasets `ChemReact` and `CovType`. `ChemReact` consists of 26733 chemicals each of feature size 100. Out of these, 2500 are test data points. The prediction variable is 0/1 and signifies if a chemical is reactive. `CovType` has 581012 webpages each of feature size 54. Out of these, 29000 are test points. The task is to predict whether a type of tree is present in each location or not.

In each of the datasets, we further randomly split the training data into 10% validation and 90% training. For the larger `CovType` data, we note that selecting about 20,000 training points out of the training set achieves about the same performance as the full set. Hence, we work with randomly selected 20,000 points for speedup. We train the logistic regression model on the new training data, and use the validation set as a proxy to the unseen test set. We build the kernel matrix $\mathbf{K}$ and the affinity vector $\mathbf{z}$, and run Algorithm 9 for various values of k. For the baselines, we use the coreset selection algorithm and random data selection as implemented by Huggins et al. [2016]. The results are presented in Figure 5.5. We note that our algorithm yields a significantly better predictive performance compared to random subsets and coresets Huggins et al. [2016] with the same size of the training subset across different subset sizes.

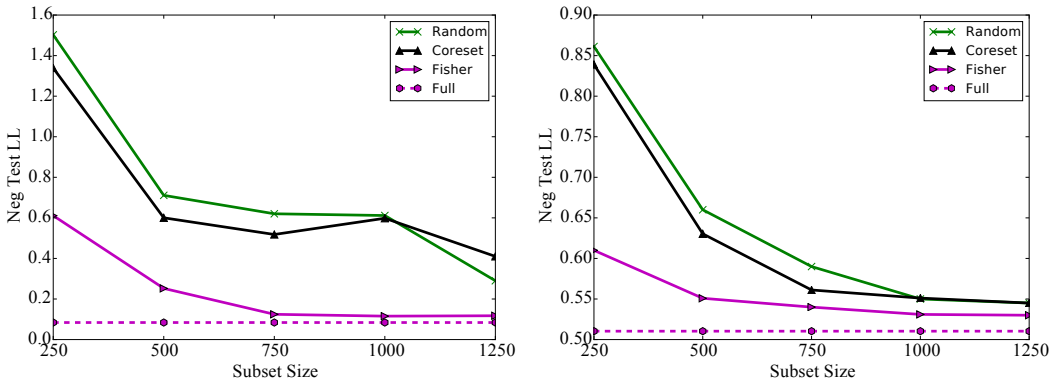


Figure 5.5: Performance for logistic regression over two datasets (left is `ChemReact` while right is `CovType`) of our method (Fisher) vs coreset selection [Huggins et al., 2016] and random data selection. 'Full' reports the numbers for training with the entire training set. Fisher achieves much better test LL performance than the baselines over several different subset sizes.

5.4.5 Qualitative Measure: Prototypes and Criticisms of Images

In this section, we learn prototypes and criticisms from the Imagenet dataset [Russakovsky et al., 2015] using image embeddings from He et al. [2015]. Each image is represented by a 2048 dimensions vector embedding, and each image belongs to one of 1000 categories. We select two breeds of one category (e.g., Blenheim spaniel) and run `MMD-critic` to learn prototypes and criticism. As shown in Figure 5.6, `MMD-critic` learns reasonable prototypes and criticisms for two types of dog breeds. On the left, criticisms picked out the different coloring (second criticism is in black and white picture), as well as pictures capturing movements of dogs (first and third criticisms). Similarly, on the right, criticisms capture the unusual, but potentially frequent pictures of dogs in costumes (first and second criticisms).

5.4.6 Quantitative measure: Prototypes and Criticisms improve interpretability

We conducted a human pilot study to collect objective and subjective measures of interpretability using `MMD-critic`. The experiment used the same dataset as Section 5.4.5. We define 'interpretability' in this work as the following: a method is interpretable if a user can correctly and efficiently predict the method's results. Under this definition, we designed a predictive task to quantitatively evaluate the interpretability. Given a randomly sampled data point, we measure how well a human can predict a group it belongs to (accuracy), and how fast they can perform the task (efficiency). We chose this dataset as the task of assigning a new image to a group requires groups to be well-explained but does not require specialized training.

We presented four conditions in the experiment. 1) raw images condition (Raw Condition) 2) Prototypes Only (Proto Only Condition) 3) Prototypes and criticisms (Proto and Criticism Condition) 4) Uniformly sampled data points per group (Uniform Condition). Raw Condition contained 100 images per species (e.g., if a group contains 2 species, there are 200 images) Proto Only Condition, Proto and Criticism Condition and Uniform Condition contains the same number of images.

We used within-subject design to minimize the effect of inter-participant variability, with a balanced Latin square to account for a potential learning effect. The four conditions were assigned to four participants (four males) in a balanced manner. Each subject answered 21 questions, where the first

Figure 5.6: Learned prototypes and criticisms from Imagenet dataset (two types of dog breeds)

three questions are practice questions and not included in the analysis. Each question showed six groups (e.g., red fox, kit fox) of a species (e.g., fox), and a randomly sampled data point that belongs to one of the groups. Subjects were encouraged to answer the questions as quickly and accurately as possible. A break was imposed after each question to mitigate the potential effect of fatigue. We measured the accuracy of answers as well as the time they took to answer each question. Participants were also asked to respond to 10 5-point Likert scale survey questions about their subjective measure of accuracy and efficiency. Each survey question compared a pair of conditions (e.g., Condition A was more helpful than condition B to correctly (or efficiently) assign the image to a group).

Subjects performed the best using Proto and Criticism Condition (M=87.5%, SD=20%). The performance with Proto Only Condition was relatively similar (M=75%, SD=41%), while that with Uniform Condition (M=55%, SD=38%, 37% decrease) and Raw Condition (M=56%, SD=33%, 36% decrease) was sub-

stantially lower. In terms of speed, subjects were most efficient using Proto Only Condition (M=1.04 mins/question, SD=0.28, 44% decrease compared to Raw Condition), followed by Uniform Condition (M=1.31 mins/question, SD=0.59) and Proto and Criticism Condition (M=1.37 mins/question, SD=0.8). Subjects spent the most time with Raw Condition (M=1.86 mins/question, SD=0.67).

Subjects indicated their preference of Proto and Criticism Condition over Raw Condition and Uniform Condition. In a survey question that asks to compare Proto and Criticism Condition and Raw Condition, a subject added that "[Proto and Criticism Condition resulted in] less confusion from trying to discover hidden patterns in a ton of images, more clues indicating what features are important". In particular, in a question that asks to compare Proto and Criticism Condition and Proto Only Condition, a subject said that "The addition of criticisms made it easier to locate the defining features of the cluster within the prototypical images". The humans' superior performance with prototypes and criticism in this preliminary study shows that providing criticisms together with prototypes is a promising direction to improve the interpretability.

Chapter 6

Weakly Submodular Functions

In this chapter, we dive deeper into weakly submodular functions, and present novel results to scale up the classic greedy algorithm and discuss an extension to the low rank case. A greedy approach to optimizing a set function is myopic – the algorithm chooses the element from the available choices that gives the largest *incremental* gain for the set of choices previously made. The algorithm is illustrated in Algorithm 9. The algorithm makes k *outer* iterations, where k is the desired sparsity. Each iteration is a full pass over the remaining candidate choices, wherein the marginal gain is calculated for each remaining candidate choice. Thus the greedy algorithm has the computational complexity of $O(dk)$ calls to the function evaluation oracle.

Algorithm 9 GREEDY(S, k)

Input: sparsity k, available choices S
$\mathsf{A}_0 = \emptyset$
for $i \in 0 \dots (k-1)$ **do**
    $s = \arg\max_{j \in \mathsf{S} \setminus \mathsf{A}_i} f(\mathsf{A}_i \cup \{j\}) - f(\mathsf{A}_i)$
    $\mathsf{A}_{i+1} = \mathsf{A}_i \cup \{s\}$
end for
return A_k

For a large d, the linear scaling of the greedy algorithm for a fixed

k may be prohibitive. As such, algorithms that scale sublinearly are useful for truly large scale selections. The DistributedGreedy algorithm, for example, achieves this sublinear scaling by making use of multiple machines. The data is split uniformly at random to l machines. Each machine then performs its own independent greedy selection (Algorithm 9), and outputs a k sized solution. All of the greedy solutions are collated by a central machine, which performs another round of the greedy selections to output the final solution. The algorithm is illustrated in Algorithm 10, and is analyzed in Section 6.1. It has a computational complexity of $O(dk/l)$ The algorithm is easy to implement in parallel or within a distributed computing framework *e.g.* MapReduce.

Algorithm 10 DistributedGreedy(l, k,$\{\mathsf{A}_j\}$)

1: Input: sparsity k, number of parallel solvers l, partition $\{\mathsf{A}_j\}$ of the set of available choices A
2: $\mathsf{G}_i \leftarrow \textsc{Greedy}(\mathsf{A}_j,\, k)\ \forall j \in [l]$
3: $\mathsf{G} \leftarrow \textsc{Greedy}(\cup_j \mathsf{G}_j,\, k)$
4: $\mathsf{G}_{\max} \leftarrow \arg\max_{\mathsf{G}_j} f(\mathsf{G}_j)$
5: return $\arg\max f(\mathsf{G}), f(\mathsf{G}_{\max})$

An alternative to distributing the data, say when several machines are not available, is to perform the greedy selection *stochastically*. The StochasticGreedy algorithm for submodular functions was introduced by Mirzasoleiman et al. [2015]. At any given iteration $i \in [k]$, instead of performing a function evaluation for each of the remaining $(d-i)$ candidates, a subset of a fixed size $C = \lceil \frac{d \log 1/\delta}{k} \rceil$ (where δ is a pre-specified hyperparameter) is uniformly sampled from the available $(d-i)$ choices using the subroutine Subsample, and

the function evaluation is made on those subsampled choices as if they were the only available candidates. This speeds up the greedy algorithm to $O(Ck)$ function evaluations. The algorithm is presented in Algorithm 11, and its approximation bounds are discussed in Section 6.2.

Algorithm 11 StochasticGreedy(S, k, δ)

Input: sparsity k, available choices S, subsampling parameter δ
$\mathsf{A}_0 = \emptyset$
for $i \in 0 \dots (k-1)$ **do**
  $\mathsf{S}_\delta \leftarrow$ Subsample($\mathsf{S} \backslash \mathsf{A}_i$, δ, k)
  $s = \arg\max_{j \in \mathsf{S}_\delta} f(\mathsf{A}_i \cup \{j\}) - f(\mathsf{A}_i)$
  $\mathsf{A}_{i+1} = \mathsf{A}_i \cup \{s\}$
end for
return A_k

Finally, we discuss a property of the greedy algorithm, which is fundamental to the analysis of the DistributedGreedy algorithm. The greedy algorithm belongs to a larger class of algorithms called 1-nice algorithms [Mirrokni and Zadimoghaddam, 2015]. The following result allows us to remove or add unselected items from the choice set that is accessible to the algorithm.

Lemma 6.0.1. *[Mirrokni and Zadimoghaddam, 2015] Say* $|\mathsf{S}| > k$*, and let* $\textsc{Greedy}(\mathsf{S}, k) \subset \mathsf{S}$ *be the* k*-sized set returned by Algorithm 9. For any* $x \notin \textsc{Greedy}(\mathsf{S}, k)$*,* $\textsc{Greedy}(\mathsf{S} \backslash \{x\}, k) = \textsc{Greedy}(\mathsf{S}, k)$*.*

Note that Lemma 6.0.1 is a property of the algorithm, and is independent of the function itself. Prior works [Mirrokni and Zadimoghaddam, 2015], [Barbosa et al., 2015] have exploited this property in conjunction with

properties of submodular functions to obtain approximation bounds for the distributed algorithms. Our work extends these results to weakly submodular functions. As such, it is easy to see that our results are easily extensible to other *nice* algorithms – including distributed OMP and distributed stochastic greedy – that have closed form bounds for the respective single machine algorithm. For ease of exposition, we focus our discussion on the distributed greedy algorithm.

For the case of greedy low rank optimization, we show that the problem falls under the purview of weakly submodular functions, even though the underlying candidate set of rank-1 matrices is infinite. We choose a matching pursuit greedy variant for efficient support selection, and derive novel approximation guarantees that improve upon existing bounds by an exponential factor.

6.1 Distributed Greedy

In this section, we obtain approximation bounds for DistributedGreedy (Algorithm 10). The algorithm returns the best out of $(l+1)$ solutions : the l *local* solutions (steps 2,4), and the final aggregated one (step 3). Our strategy to obtain the approximation bound for the algorithm is as follows. To obtain an overall approximation bound, we obtain individual bounds on each of the solutions in terms of the submodularity ratio (Definition 2.1.7) and use the subadditivity ratio (Definition 6.1.1) to show that one of the two shall always hold. For approximation bounds on the *local* solutions, we make use of the

niceness of the GREEDY (Lemma 6.0.1). The bound on the aggregated solution is more involved, since it involves tracking the split of the true solution $\mathsf{A}^\star$ across machines. The assumption of partitioning uniformly at random is vital here. This helps us lower bound the greedy gain in expectation by a probabilistic overlap with the true solution. The trick of tracking the split of the true solution across machines is similar to the one that has been used for analysis of submodular functions [Mirrokni and Zadimoghaddam, 2015], [Barbosa et al., 2015], but without the explicit connection to submodularity and subadditivity ratios.

We next define the subadditivity ratio, which helps us generalize subadditive functions in the way similar to how submodularity ratio generalizes submodular functions.

Definition 6.1.1 (Subaddivity ratio)**.** We define the subadditivity ratio for a set function f w.r.t a set S as:

$$\nu_{\mathsf{S}} := \min_{\substack{\mathsf{A}\cup\mathsf{B}=\mathsf{S} \\ \mathsf{A}\cap\mathsf{B}=\phi}} \frac{f(\mathsf{A}) + f(\mathsf{B})}{f(\mathsf{S})}.$$

We further define the subadditivity ratio of a function for an integer k, ν_k, which takes a uniform bound over all sets of size k:

$$\nu_k := \min_{\mathsf{S}:|\mathsf{S}|=k} \nu_{\mathsf{S}}.$$

By definition, the function $f(\cdot)$ is subadditive iff $\nu_{\mathsf{S}} \geq 1, \forall \mathsf{S} \subset [d]$. Since submodularaity implies subadditivity (the converse is not always true), if the function $f(\cdot)$ is submodular, $\nu_{\mathsf{S}} \geq 1, \forall \mathsf{S} \subset [d]$.

We next present some notation and few lemmas that lead up to the main result of this section (Theorem 6.1.3). Let A be the entire set of available choices. Partition the set A uniformly at random into $\mathsf{A}_1, \ldots, \mathsf{A}_l$. Let G_j be the k-sized solution returned by running the greedy algorithm on A_j *i.e.* $\mathsf{G}_j = \textsc{Greedy}(\mathsf{A}_j, k)$. Note that each A_j induces a partition onto the optimal k-sized solution A^* as follows:

$$\mathsf{S}_j := \{x \in \mathsf{A}^* : x \in \textsc{Greedy}(\mathsf{A}_j \cup x, k)\},$$
$$\mathsf{T}_j := \{x \in \mathsf{A}^* : x \notin \textsc{Greedy}(\mathsf{A}_j \cup x, k)\}.$$

Having defined the notation, we start by lower bounding the local solutions in terms of value of the subset of $\mathsf{A}^\star$ that is not selected as part of the respective local solution.

Lemma 6.1.1. $f(\mathsf{G}_j) \geq (1 - \exp(-\gamma_{\mathsf{G}_j,k}))f(\mathsf{T}_j)$.

The next lemma is used to lower the bound the value of the aggregated solution (step 4 in Algorithm 10) in terms of the value of the subset $\mathsf{A}^\star$ that is selected as part of the respective local solution.

Lemma 6.1.2. $\exists j \in [l]$ *s.t.* $\mathbb{E}[f(\mathsf{G})] \geq \left(1 - \frac{1}{e^{\gamma_{\mathsf{G},k}}}\right) f(\mathsf{S}_j)$.

We are now ready to present our main result about the approximation guarantee for Algorithm 10.

Theorem 6.1.3. *Let G_{dg} be the set returned by the distributed greedy (Algorithm 10). Let $\gamma = \min\{\gamma_{\mathsf{G}_i,k}, \gamma_{\mathsf{G},k}\}$. Then,*

$$\mathbb{E}[f(\mathsf{G}_{dg})] \geq \frac{\nu_k}{2}\left(1 - \exp(-\gamma)\right) f(\mathsf{A}^*). \tag{6.1}$$

Proof. There are l machines, each with its local greedy solution $\mathsf{G}_i, i \in [l]$. In addition, there is the aggregated solution set G. The key idea is to show that atleast one of the $(l+1)$ solutions is *good enough*.

Say $f(\mathsf{T}_i) \geq \frac{\nu_k}{2} f(\mathsf{A}^*)$ for some i, then by Lemma 6.1.1, $f(\mathsf{G}_i) \geq \frac{\nu_k}{2}(1 - \exp(-\gamma_{\mathsf{G}_i,k}))f(\mathsf{A}^*)$.

On the other hand, say for all i, $f(\mathsf{T}_i) < \frac{\nu_k}{2} f(\mathsf{A}^*)$, then for all i, $f(\mathsf{S}_i) > \frac{\nu_k}{2} f(\mathsf{A}^*)$. By Lemma 6.1.2, the result then follows. □

Theorem 6.2.2 generalizes the approximation guarantee of $\frac{1}{2}(1 - \frac{1}{e})$ obtained by Barbosa et al. [2015] for submodular functions. Their analysis uses convexity of the Lovasz extension of submodular functions, and hence can not be trivially extended to weakly submodular functions. In addition to being applicable for a larger class of functions, our result can also provide tighter bounds for specific applications or datasets even for submodular functions, since they are also applicable for submodular functions, and bounding ν_k and γ away from 1 from domain knowledge will give tighter approximations than the generic bound of $\frac{1}{2}(1 - \frac{1}{e})$.

6.2 Stochastic Greedy

For analysis of Algorithm 11, we show that the subsampling parameter δ governs the tradeoff between the speedup and the loss in approximation

quality *vis-a-vis* the classic GREEDY. Before formally providing the approximation bound, we present an auxillary lemma that is key to proving the new approximation bound. The following result is a generalization of Lemma 2 from Mirzasoleiman et al. [2015] for weakly submodular functions.

Lemma 6.2.1. *Let* $\mathsf{A}, \mathsf{B} \subset [n]$*, with* $|\mathsf{B}| \leq k$*. Consider another set* C *drawn randomly from* $[n] \backslash \mathsf{A}$ *with* $|\mathsf{C}| = \lceil \frac{n \log 1/\delta}{k} \rceil$*. Then,*

$$\mathbb{E}[\max_{v \in \mathsf{C}} f(v \cup \mathsf{A}) - f(\mathsf{A})] \geq \frac{(1-\delta)\gamma_{\mathsf{A},\mathsf{B}\backslash\mathsf{A}}}{k}(f(\mathsf{B}) - f(\mathsf{A})).$$

We are now ready to present our result that shows that stochastic greedy selections (Algorithm 11) can be applied to weakly submodular functions with provable approximation guarantees. All the proofs missing from the main text are presented in the supplement.

Theorem 6.2.2. *Let* $\mathsf{A}^\star$ *be the optimum set of size* k*, and* $\mathsf{A}_i = \{a_1, a_2, \ldots, a_i\}, i \leq k$ *be the set built by* STOCHASTICGREEDY *at step* i*. Then,* $\mathbb{E}[f(\mathsf{A}_k)] \geq \left(1 - \frac{1}{e^{\gamma_{\mathsf{A}_k,k}}} - \delta\right) f(\mathsf{A}^*)$.

Proof. Define $g_i := f(\mathsf{A}_i) - f(\mathsf{A}_{i-1})$. Using Lemma 6.2.1 with $\mathsf{B} = \mathsf{A}^\star$ and $\gamma_{\mathsf{A}_{i-1},\mathsf{B}\backslash\mathsf{A}} \geq \gamma_{\mathsf{A}_k,k}$, we get at the i-th step,

$$\begin{aligned} \mathbb{E}[g_i | \mathsf{A}_{i-1} = \mathsf{A}] \geq & \quad \frac{(1-\delta)\gamma_{\mathsf{A}_k,k}}{k}(f(\mathsf{B}) - f(\mathsf{A})) \\ \geq & \quad \frac{(1-\delta)\gamma_{\mathsf{A}_k,k}}{k}(f(\mathsf{A}^\star) - f(\mathsf{A})). \end{aligned} \tag{6.2}$$

Define $h_{i-1} := \mathbb{E}[f(\mathsf{A}^\star) - f(\mathsf{A}_{i-1})]$, $C := \frac{(1-\delta)\gamma_{\mathsf{A}_k,k}}{k}$. Note that $\mathbb{E}[g_i] = h_{i-1} - h_i$. Taking expectation on both sides over A_i, (6.2) becomes

$$h_i \leq (1-C)h_{i-1} \leq (1-C)^i h_0.$$

Using $h_k = \mathbb{E}[f(\mathsf{A}^\star) - f(\mathsf{A}_k)]$ and $h_0 = f(\mathsf{A}^\star)$ above, along with the fact that $1 + a\delta \geq a^\delta$ for $0 \leq \delta \leq 1$,

$$\begin{aligned}
\mathbb{E}[f(\mathsf{A}_k)] &\geq \left(1 - \left(1 - \frac{(1-\delta)\gamma_{\mathsf{A}_k,k}}{k}\right)^k\right) f(\mathsf{A}^*) \\
&\geq \left(1 - \exp\left(-\gamma_{\mathsf{A}_k,k}\,(1-\delta)\right)\right) f(\mathsf{A}^*) \\
&\geq \left(1 - \frac{1}{e^{\gamma_{\mathsf{A}_k,k}}} - \delta\right) f(\mathsf{A}^*). \quad \square
\end{aligned}$$

Note that δ is the tradeoff hyperparameter between the speedup achieved by subsampling and the corresponding approximation guarantee. A larger value of δ means the algorithm is faster with weaker guarantees and vice versa. As $\delta \to 0$, we tend towards the bound $(1 - \frac{1}{e^{\gamma_{\mathsf{A}_k,k}}})$ which recovers the bound for weakly submodular functions obtained by Das and Kempe [2011] for the classic greedy selections (Algorithm 9), and also recovers the well known bound of $(1 - \frac{1}{e})$ for submodular functions.

6.3 Greedy Low Rank Optimization

In this section, we delineate our setup of low rank estimation. In order to connect to the weak submodular maximization framework more easily, we operate in the setting of maximization of a concave matrix variate function

under a low rank constraint. This is equivalent to minimizing a convex matrix variate function under the low rank constraint as considered by Shalev-Shwartz et al. [2011] or under nuclear norm constraint or regularization as considered by Jaggi and Sulovský [2010]. The goal is to maximize a function $\ell : \mathbb{R}^{n \times d} \to \mathbb{R}$:

$$\max_{\text{rank}(\mathbf{X}) \leq r} \ell(\mathbf{X}). \tag{6.3}$$

Instead of using a convex relaxation of (6.3), our approach is to enforce the rank constraint directly by adding rank 1 matrices greedily until $\mathbf{X}$ is of rank k. The rank 1 matrices to be added are obtained as outer product of vectors from the given vector sets $\mathcal{U}$ and $\mathcal{V}$. While our results hold for general vector sets $\mathcal{U}, \mathcal{V}$ assuming an oracle access to subroutines `GreedySel` and `OMPSel` (to be detailed later), for the rest of the paper we focus on the case of norm 1 balls $\mathcal{U} := \{\mathbf{x} \in \mathbb{R}^n \, s.t. \, \|\mathbf{x}\|_2 = 1\}$ and $\mathcal{V} := \{\mathbf{x} \in \mathbb{R}^d \, s.t. \, \|\mathbf{x}\|_2 = 1\}$.

The problem (6.3) can be interpreted in the context of sparsity assuming $\mathcal{U}$ and $\mathcal{V}$ are enumerable. For example, by the SVD theorem, it is known that we can rewrite $\mathbf{X}$ as $\sum_{i=1}^{k} \alpha_i \mathbf{u}_i \mathbf{v}_i^\top$, where $\forall i$, $\mathbf{u}_i \in \mathcal{U}$ and $\mathbf{v}_i \in \mathcal{V}$. By enumerating $\mathcal{U}$ and $\mathcal{V}$ under a finite precision representation of real values, one can rethink of the optimization (6.3) as finding a sparse solution for the infinite dimensional vector $\boldsymbol{\alpha}$ [Shalev-Shwartz et al., 2011, Dudik et al., 2012]. We can also optimize over support sets, similar to the classical setting of support selection for sparse vectors. For a *specified* support set L consisting of vectors from $\mathcal{U}$ and $\mathcal{V}$, let $\mathbf{U}_\mathsf{L}$ and $\mathbf{V}_\mathsf{L}$ be the matrices formed by stacking the chosen elements of $\mathcal{U}$ and $\mathcal{V}$ respectively. We define the following set function

to maximize $\ell(\cdot)$ given L.

$$f(\mathsf{L}) = \max_{\mathbf{H} \in \mathbb{R}^{|\mathsf{L}| \times |\mathsf{L}|}} \ell(\mathbf{U}_{\mathsf{L}}^{\top} \mathbf{H} \mathbf{V}_{\mathsf{L}}) - \ell(\mathbf{0}). \tag{6.4}$$

We will denote the optimizing matrix for a support set L as $\mathbf{B}^{(\mathsf{L})}$. In other words, letting $\hat{\mathbf{H}}_{\mathsf{L}}$ be the argmax obtained in (6.4), then $\mathbf{B}^{(\mathsf{L})} := \mathbf{U}_{\mathsf{L}}^{\top} \hat{\mathbf{H}}_{\mathsf{L}} \mathbf{V}_{\mathsf{L}}$. Thus, the low rank matrix estimation problem (6.3) can be reinterpreted as the following equivalent combinatorial optimization problem:

$$\max_{|\mathsf{S}| \leq k} f(\mathsf{S}). \tag{6.5}$$

6.3.1 Algorithms

Our greedy algorithm, illustrated in Algorithm 12, builds the support set incrementally – adding rank 1 matrices one at a time such that at iteration i for $1 \leq i \leq k$ the size of the chosen support set (and hence rank of the current iterate) is i. We assume access to a subroutine `GreedySel` for the greedy selection (Step 4). This subroutine solves an inner optimization problem by calling a subroutine `GreedySel` which returns an atom s from the candidate support set that ensures

$$f(\mathsf{S}_{i-1}^{G} \cup \{s\}) - f(\mathsf{S}_{i-1}^{G}) \geq \tau \left(f(\mathsf{S}_{i-1}^{G} \cup \{s^{\star}\}) - f(\mathsf{S}_{i-1}^{G}) \right),$$

where

$$s^{\star} \leftarrow \operatorname*{arg\,max}_{a \in (\mathcal{U} \times \mathcal{V}) \perp \mathsf{S}_{i-1}^{G}} f(\mathsf{S}_{i-1}^{G} \cup \{a\}) - f(\mathsf{S}_{i-1}^{G}).$$

In words, the subroutine `GreedySel` ensures that the gain in $f(\cdot)$ obtained by using the selected atom is within $\tau \in (0, 1]$ multiplicative approximation to the atom with the best possible gain in $f(\cdot)$. The hyperparameter τ governs a tradeoff allowing a compromise in myopic gain for a possibly quicker selection.

The greedy selection requires fitting and scoring every candidate support, which is often prohibitively expensive. An alternative is to choose the next atom by using the linear maximization oracle used by Frank-Wolfe [Jaggi, 2013] or Matching Pursuit algorithms [Gribonval and Vandergheynst, 2006, Locatello et al., 2017]. This step replaces Step 4 of Algorithm 12 as illustrated in Algorithm 13. Let $\mathsf{L} = \mathsf{S}^O_{i-1}$ be the set constructed by the algorithm at iteration $(i-1)$. The linear oracle `OMPSel` returns an atom s for iteration i ensuring

$$\langle \nabla \ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{u}_s \mathbf{v}_s^\top \rangle \geq \tau \max_{(\mathbf{u},\mathbf{v}) \in (\mathcal{U} \times \mathcal{V}) \perp \mathsf{S}^O_{i-1}} \langle \nabla \ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{u}\mathbf{v}^\top \rangle.$$

The linear problem `OMPSel` can be considerably faster that `GreedySel`. `OMPSel` reduces to finding the left and right singular vectors of $\nabla \ell(\mathbf{B}^{(\mathsf{L})})$ corresponding to its largest singular value. If t is the number of non-zero entries in $\nabla \ell(\mathbf{B}^{(\mathsf{L})})$, then this takes $O(\frac{t}{1-\tau}(\log n + \log d))$ time.

We note that Algorithm 13 is the same as considered by Shalev-Shwartz et al. [2011] as GECO (Greedy Efficient Component Optimization). However, as we shall see, our analysis provides stronger bounds than their Theorem 2.

Algorithm 12 GREEDY($\mathcal{U}$, $\mathcal{V}$, k, τ)

1: **Input:** vector sets $\mathcal{U}$, $\mathcal{V}$, sparsity parameter k, subroutine hyperparameter τ
2: $\mathsf{S}_0^G \leftarrow \emptyset$
3: **for** $i = 1 \dots k$ **do**
4: $\quad s \leftarrow \texttt{GreedySel}(\tau)$
5: $\quad \mathsf{S}_i^G \leftarrow \mathsf{S}_{i-1}^G \cup \{s\}$
6: **end for**
7: **return** S_k^G, $\mathbf{B}^{(\mathsf{S}_k^G)}$, $f(\mathsf{S}_k^G)$.

Algorithm 13 GECO($\mathcal{U}$, $\mathcal{V}$, k, τ)

same as Algorithm 12 except
4: $s \leftarrow \texttt{OMPSel}(\tau)$

Remark 6.3.1. We note that Step 5 of Algorithms 12 and 13 requires solving the RHS of (6.4) which is a matrix variate problem of size i^2 at iteration i. This refitting is equivalent to the "fully-corrective" versions of Frank-Wolfe/Matching Pursuit algorithms [Locatello et al., 2017, Lacoste-Julien and Jaggi, 2015] which, intuitively speaking, extract out all the information w.r.t $\ell(\cdot)$ from the chosen set of atoms, thereby ensuring that the next rank 1 atom chosen has row and column space orthogonal to the previously chosen atoms. Thus the constrained maximization on the orthogonal complement of S_i^O in subroutine `OMPSel` (S_i^G in `GreedySel`) need not be explicitly enforced, but is still shown for clarity.

6.3.2 Analysis

In this section, we prove that low rank matrix optimization over the rank one atoms satisfies weak submodularity. We explicitly delineate some notation and assumptions. With slight abuse of notation, we assume $\ell(\cdot)$ is m_i-strongly concave and M_i-smooth over pairs of matrices of rank i. For $i \leq j$, note that $m_i \geq m_j$ and $M_i \leq M_j$. Additionally, let $\tilde{\Omega} := \{(\mathbf{X}, \mathbf{Y}) : \text{rank}(\mathbf{X} - \mathbf{Y}) \leq 1\}$, and assume $\ell(\cdot)$ is $\tilde{M}_1$-smooth over $\tilde{\Omega}$. It is easy to see $\tilde{M}_1 \leq M_1$.

First we prove that if the low rank RSC holds (Definition 2.1.8), then the submodularity ratio (Definition 2.1.7) is lower-bounded by the inverse condition number.

Theorem 6.3.1. *Let* L *be a set of* k *rank* 1 *atoms and* S *be a set of* r *rank* 1 *atoms where we sequentially orthogonalize the atoms against* L*. If* $\ell(\cdot)$ *is* m_i*-strongly concave over matrices of rank* i*, and* $\tilde{M}_1$*-smooth over the set* $\tilde{\Omega} := \{(\mathbf{X}, \mathbf{Y}) : \textit{rank}(\mathbf{X} - \mathbf{Y}) = 1\}$*, then*

$$\gamma_{\mathsf{L},r} := \frac{\sum_{a \in S}[f(\mathsf{L} \cup \{a\}) - f(\mathsf{L})]}{f(\mathsf{L} \cup \mathsf{S}) - f(\mathsf{L})} \geq \frac{m_{r+k}}{\tilde{M}_1}.$$

Proof. An important aspect of the assumptions is that the space of atoms spanned by S is orthogonal to the span of L. Furthermore, $\text{span}(\mathsf{L} \cup \mathsf{S}) \supset \text{span}(\mathsf{S})$. Let $\bar{k} = k + r$. We will first upper bound the denominator in the submodularity ratio. From strong concavity,

$$\frac{m_{\bar{k}}}{2}\|\mathbf{B}^{(\mathsf{L}\cup\mathsf{S})} - \mathbf{B}^{(\mathsf{L})}\|_F^2 \leq \ell(\mathbf{B}^{(\mathsf{L})}) - \ell(\mathbf{B}^{(\mathsf{L}\cup\mathsf{S})}) + \langle \nabla \ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}^{(\mathsf{L}\cup\mathsf{S})} - \mathbf{B}^{(\mathsf{L})} \rangle$$

Rearranging

$$
\begin{aligned}
0 \leq \ell(\mathbf{B}^{(\mathsf{L}\cup\mathsf{S})}) - \ell(\mathbf{B}^{(\mathsf{L})}) &\leq \langle \nabla \ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}^{(\mathsf{L}\cup\mathsf{S})} - \mathbf{B}^{(\mathsf{L})} \rangle - \frac{m_{\bar{k}}}{2} \|\mathbf{B}^{(\mathsf{L}\cup\mathsf{S})} - \mathbf{B}^{(\mathsf{L})}\|_F^2 \\
&\leq \underset{\substack{\mathbf{X}: \\ \mathbf{X}=\mathbf{U}_{\mathsf{L}\cup\mathsf{S}}\mathbf{H}\mathbf{V}_{\mathsf{L}\cup\mathsf{S}} \\ \mathbf{H}\in\mathbb{R}^{|\mathsf{L}\cup\mathsf{S}|\times|\mathsf{L}\cup\mathsf{S}|}}}{\arg\max} \langle \nabla \ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{X} - \mathbf{B}^{(\mathsf{L})} \rangle - \frac{m_{\bar{k}}}{2} \|\mathbf{X} - \mathbf{B}^{(\mathsf{L})}\|_F^2 \\
&= \underset{\substack{\mathbf{X}: \\ \mathbf{X}=\mathbf{U}_{\mathsf{L}\cup\mathsf{S}}\mathbf{H}\mathbf{V}_{\mathsf{L}\cup\mathsf{S}} \\ \mathbf{H}\in\mathbb{R}^{|\mathsf{L}\cup\mathsf{S}|\times|\mathsf{L}\cup\mathsf{S}|}}}{\arg\max} \langle P_{\mathbf{U}_{\mathsf{S}}}(\nabla \ell(\mathbf{B}^{(\mathsf{L})}))P_{\mathbf{V}_{\mathsf{S}}}, \mathbf{X} - \mathbf{B}^{(\mathsf{L})} \rangle - \frac{m_{\bar{k}}}{2} \|\mathbf{X} - \mathbf{B}^{(\mathsf{L})}\|_F^2,
\end{aligned}
$$

where the last equality holds because $\langle (\nabla \ell(\mathbf{B}^{(\mathsf{L})})), P_{\mathbf{U}_{\mathsf{L}}} \mathbf{X} P_{\mathbf{V}_{\mathsf{L}}} - \mathbf{B}^{(\mathsf{L})} \rangle = 0$. Solving the argmax problem, we get $\mathbf{X} = \mathbf{B}^{(\mathsf{L})} + \frac{1}{m_{\bar{k}}} P_{\mathbf{U}_{\mathsf{S}}}(\nabla \ell(\mathbf{B}^{(\mathsf{L})}))P_{\mathbf{V}_{\mathsf{S}}}$. Plugging in, we get,

$$
\ell(\mathbf{B}^{(\mathsf{L}\cup\mathsf{S})}) - \ell(\mathbf{B}^{(\mathsf{L})}) \leq \frac{1}{2m_{\bar{k}}} \|P_{\mathbf{U}_{\mathsf{S}}}(\nabla \ell(\mathbf{B}^{(\mathsf{L})}))P_{\mathbf{V}_{\mathsf{S}}}\|_F^2
$$

We next bound the numerator. Recall that the atoms in S are orthogonal to each other *i.e.* $\mathbf{U}_{\mathsf{S}}$ and $\mathbf{V}_{\mathsf{S}}$ are both orthonormal.

For clarity, we define the shorthand, $\mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})} = \langle \mathbf{u}_i \mathbf{v}_j^\top, \mathbf{B}^{(\mathsf{L}\cup\mathsf{S})} \rangle \, \mathbf{u}_i \mathbf{v}_j^\top$, for $i, j \in [|\mathsf{L} \cup \mathsf{S}|]$.

With an arbitrary $i \in \mathsf{S}$, and arbitrary scalars $\alpha_{ii}, \alpha_{ij}, \alpha_{ji}$ for $j \in \mathsf{L}$,

$$
\begin{aligned}
\ell(\mathbf{B}^{(\mathsf{L}\cup\{i\})}) - \ell(\mathbf{B}^{(\mathsf{L})}) &\geq \ell(\mathbf{B}^{(\mathsf{L})} + \alpha_{ii}\mathbf{B}_{ii}^{(\mathsf{L}\cup\mathsf{S})} + \sum_{j\in\mathsf{L}} \alpha_{ij}\mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})} + \sum_{j\in\mathsf{L}} \alpha_{ji}\mathbf{B}_{ji}^{(\mathsf{L}\cup\mathsf{S})}) - \ell(\mathbf{B}^{(\mathsf{L})}) \\
&\geq \langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \alpha_{ii}\mathbf{B}_{ii}^{(\mathsf{L}\cup\mathsf{S})} + \sum_{j\in\mathsf{L}} \alpha_{ij}\mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})} + \sum_{j\in\mathsf{L}} \alpha_{ji}\mathbf{B}_{ji}^{(\mathsf{L}\cup\mathsf{S})} \rangle \\
&\quad - \frac{\tilde{M}_1}{2}\left[\alpha_{ii}^2\|\mathbf{B}_{ii}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2 + \sum_{j\in\mathsf{L}} \alpha_{ij}^2\|\mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2 + \sum_{j\in\mathsf{L}} \alpha_{ji}^2\|\mathbf{B}_{ji}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2\right]. \\
&\geq \frac{\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}_{ii}^{(\mathsf{L}\cup\mathsf{S})}\rangle^2}{2\tilde{M}_1\|\mathbf{B}_{ii}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2} \\
&\quad + \sum_{j\in\mathsf{L}}\left(\frac{\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})}\rangle^2}{2\tilde{M}_1\|\mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2} + \frac{\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}_{ji}^{(\mathsf{L}\cup\mathsf{S})}\rangle^2}{2\tilde{M}_1\|\mathbf{B}_{ji}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2}\right),
\end{aligned}
$$

where the last inequality follows by setting $\alpha_{ij} = \frac{\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})}\rangle}{\tilde{M}_1\|\mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2}$ for $j \in \mathsf{L}$, and for $j = i$.

Summing up for all $i \in \mathsf{S}$, we get

$$
\begin{aligned}
\sum_{i\in\mathsf{S}} \ell(\mathbf{B}^{(\mathsf{L}\cup\{i\})}) - \ell(\mathbf{B}^{(\mathsf{L})}) &\geq \sum_{i\in\mathsf{S}} \frac{\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}_{ii}^{(\mathsf{L}\cup\mathsf{S})}\rangle^2}{2\tilde{M}_1\|\mathbf{B}_{ii}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2} \\
&\quad + \sum_{i\in\mathsf{S}}\sum_{j\in\mathsf{L}}\left(\frac{\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})}\rangle^2}{2\tilde{M}_1\|\mathbf{B}_{ij}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2} + \frac{\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{B}_{ji}^{(\mathsf{L}\cup\mathsf{S})}\rangle^2}{2\tilde{M}_1\|\mathbf{B}_{ji}^{(\mathsf{L}\cup\mathsf{S})}\|_F^2}\right) \\
&= \frac{1}{2\tilde{M}_1}\|P_{\mathbf{U}_\mathsf{S}}\nabla\ell(\mathbf{B}^{(\mathsf{L})})P_{\mathbf{V}_\mathsf{S}}\|_F^2
\end{aligned}
$$

□

The proof of Theorem 6.3.1 is structured around individually obtaining a lower bound for the numerator and an upper bound for the denominator of the submodularity ratio by exploiting the concavity and convexity conditions.

6.3.2.1 Greedy Improvement

Bounding the submodularity ratio is crucial to obtaining approximation guarantees for Algorithm 2.

Theorem 6.3.2. *Let* $\mathsf{S} := \mathsf{S}_k^G$ *be the greedy solution set obtained by running Algorithm 2 for* k *iterations, and let* $\mathsf{S}^\star$ *be an optimal support set of size* r*. Let* $\ell(\cdot)$ *be* m_i *strongly concave on the set of matrices with rank less than or equal to* i*, and* $\tilde{M}_1$ *smooth on the set of matrices in the set* $\tilde{\Omega}$*. Then,*

$$f(\mathsf{S}) \geq \left(1 - \frac{1}{e^{c_1}}\right) f(\mathsf{S}^\star) \geq \left(1 - \frac{1}{e^{c_2}}\right) f(\mathsf{S}^\star),$$

where $c_1 = \tau\gamma_{\mathsf{S},r}\frac{k}{r}$ *and* $c_2 = \tau\frac{m_{r+k}}{\tilde{M}_1}\frac{k}{r}$*.*

We now provide the proof of Theorem 6.3.2. Let S_i^G be the support set formed by Algorithm 2 at iteration i. Define $A(i) := f(\mathsf{S}_i^G) - f(\mathsf{S}_{i-1}^G)$ with $A(0) = 0$ as the greedy improvement. We also define $B(i) := f(\mathsf{S}^*) - f(\mathsf{S}_i^G)$ to be the remaining amount to improve, where $\mathsf{S}^\star$ is the optimum k-sized solution. We provide an auxiliary Lemma that uses the submodularity ratio to lower bound the greedy improvement in terms of best possible improvement from step i.

Lemma 6.3.3. *At iteration* i*, the incremental gain of the greedy method (Algorithm 2) is*

$$A(i+1) \geq \frac{\tau\gamma_{\mathsf{S}_i^G,r}}{r} B(i).$$

Proof. Let $\mathsf{S} = \mathsf{S}_i^G$. Let S^R be the sequential orthogonalization of the atoms in S^* relative to S. Thus,

$$\begin{aligned} rA(i+1) \geq |\mathsf{S}^R| A(i+1) &\geq \tau |\mathsf{S}^R| \max_{a \in \mathsf{S}^R} f(\mathsf{S} \cup \{a\}) - f(\mathsf{S}) \\ &\geq \tau \sum_{a \in \mathsf{S}^R} [f(\mathsf{S} \cup \{a\}) - f(\mathsf{S})] \\ &\geq \tau \gamma_{\mathsf{S},|\mathsf{S}^R|} [f(\mathsf{S} \cup \mathsf{S}^R) - f(\mathsf{S})] \\ &\geq \tau \gamma_{\mathsf{S},|\mathsf{S}^R|} B(i) \end{aligned}$$

Note that the last inequality follows because $f(\mathsf{S} \cup \mathsf{S}^R) \geq f(\mathsf{S}^*)$. The penultimate inequality follows by the definition of weak submodularity, which applies in this case because the atoms in S^R are orthogonal to eachother and are also orthogonal to S. □

Using Lemma 6.3.3, one can prove an approximation guarantee for Algorithm 2.

Proof of Theorem 6.3.2

Proof. From the notation used for Lemma 6.3.3, $A(i+1) = B(i) - B(i+1)$. Let $C = \frac{\tau \gamma_{\mathsf{S}_i^G, r}}{r}$. From Lemma 6.3.3, we have,

$$B(i+1) \leq (1-C)B(i) \leq (1-C)^{i+1} B(0).$$

From its definition, $B(0) = f(\mathsf{S}^\star) - f(\emptyset)$. So we get,

$$
\begin{aligned}
&[f(\mathsf{S}^\star) - f(\emptyset)] - \left[f(\mathsf{S}_i^G) - f(\emptyset)\right] \leq (1-C)^i \left[f(\mathsf{S}^\star) - f(\emptyset)\right] \\
\implies &\left[f(\mathsf{S}_i^G) - f(\emptyset)\right] \geq \left(1 - (1-C)^i\right) \left[f(\mathsf{S}^\star) - f(\emptyset)\right] \\
&\geq \left(1 - \frac{1}{e^{\tau \gamma_{\mathsf{S}_i^G, r} \frac{k}{r}}}\right) \left[f(\mathsf{S}^\star) - f(\emptyset)\right]
\end{aligned}
$$

from which the result follows. □

The proof technique for the first inequality of Theorem 6.3.2 relies on lower bounding the progress made in each iteration of Algorithm 2. Intuitively, it exploits weak submodularity to make sure that each iteration makes *enough* progress, and then applies an induction argument for r iterations. We also emphasize that the bounds in Theorem 6.3.2 are for *normalized* set function $f(\cdot)$ (which means $f(\emptyset) = 0$). A more detailed proof is presented in the appendix.

The bounds obtained in Theorem 6.3.2 are similar to the one obtained in submodular maximization of monotone normalized functions [Nemhauser et al., 1978]. In fact, our result can be re-interpreted as an extension to previous results. The greedy algorithm for submodular maximization assumes finite ground sets. We extend this for infinite ground sets. We can do this (for matrices) as long as we have an implementation of the oracle `GreedySel`. Once the choice is made by the oracle, standard analysis holds.

Remark 6.3.2. Theorem 6.3.2 provides the approximation guarantees for running the greedy selection algorithm up to k iterations to obtain a rank k matrix

iterate *vis-a-vis* the best rank r approximation. For $r = k$, and $\tau = 1$, we get an approximation bound $(1 - e^{-m/M})$ which is reminiscent of the greedy bound of $(1 - 1/e)$ under the framework of submodularity. Note that our analysis can not be used to establish classical submodularity. However, establishing weak submodularity that lower bounds γ is sufficient to provide slightly weaker than classical submodularity guarantees.

Remark 6.3.3. Theorem 6.3.2 implies that to obtain $(1-\epsilon)$ approximation guarantee in the worst case, running Algorithm 2 for $k = \frac{rM}{m\tau} \log \frac{1}{\epsilon}) = O(r \log 1/\epsilon)$ iterations suffices. This is useful when the application allows a tradeoff: compromising on the low rank constraint a little to achieve tighter approximation guarantees.

Remark 6.3.4. Das and Kempe [2011] considered the special case of greedily maximizing R^2 statistic for linear regression, which corresponds to classical sparsity in vectors. They also obtain a bound of $(1 - 1/e^{\gamma})$, where γ is the submodularity ratio for their respective setup. This was generalized by Elenberg et al. [2018] to general concave functions under sparsity constraints. Our analysis is for the low rank constraint, as opposed to sparsity in vectors that was considered by them.

6.3.2.2 GECO Improvement

In this section, we obtain approximation guarantees for Algorithm 13. The greedy search over the infinitely many candidate atoms is infeasible, especially when $\tau = 1$. Thus while Algorithm 2 establishes interesting theoretical

connections with submodularity, it is not practical in general. To obtain a tractable and practically useful algorithm, the greedy search is replaced by a Frank Wolfe or Matching Pursuit style linear optimization which can be easily implemented as finding the top singular vectors of the gradient at iteration i. In this section, we show that despite the speedup, we lose very little in terms of approximation guarantees. In fact, if the approximation factor τ in `OMPSel`() is 1, we get the same bounds as those obtained for the greedy algorithm.

Theorem 6.3.4. *Let* $\mathsf{S} := \mathsf{S}_k^O$ *be the greedy solution set obtained using Algorithm 13 for* k *iterations, and let* $\mathsf{S}^\star$ *be the optimum size* r *support set. Let* $\ell(\cdot)$ *be* m_{r+k} *strongly concave on the set of matrices with rank less than or equal to* $(r+k)$*, and* $\tilde{M}_1$ *smooth on the set of matrices with rank in the set* $\tilde{\Omega}$*. Then,*

$$f(\mathsf{S}) \geq \left(1 - \frac{1}{e^{c_3}}\right) f(\mathsf{S}^\star),$$

where $c_3 = \tau^2 \frac{m_{r+k}}{\tilde{M}_1} \frac{k}{r}$.

The proof of Theorem 6.3.4 follows along the lines of Theorem 6.3.2. The central idea is similar – to exploit the RSC conditions to make sure that each iteration makes *sufficient* progress, and then provide an induction argument for r iterations. Unlike the greedy algorithm, however, using the submodularity ratio is no longer required. Note that the bound obtained in Theorem 6.3.4 is similar to Theorem 6.3.2, except the exponent on the approximation factor τ.

Let S_i^O be the support set selected by the GECO procedure (Algorithm 13) at iteration i. Similar to the section on greedy improvement, we

define some notation. Let $D(i) := f(\mathsf{S}_i^O) - f(\mathsf{S}_{i-1}^O)$ be the improvement made at step i, and as before we have $B(i) = f(\mathsf{S}^\star) - f(\mathsf{S}_i^O)$ be the remaining amount to improve.

We prove the following auxiliary lemma which lower bounds the gain after adding the atom selected by the subroutine `OMPSel`in terms of operator norm of the gradient of the current iterate and smoothness of the function.

Lemma 6.3.5. *Assume that $\ell(\cdot)$ is m_i-strongly concave and M_i-smooth over matrices of in the set $\tilde{\Omega} := \{(\mathbf{X}, \mathbf{Y}) : \mathit{rank}(\mathbf{X} - \mathbf{Y}) \leq 1\}$. Then,*

$$D(i+1) \geq \frac{\tau m_{r+k}}{r\tilde{M}_1} B(i).$$

Proof. For simplicity, say $\mathsf{L} = \mathsf{S}_i^O$. Recall that for a given support set L, $f(\mathsf{L}) = \ell(\mathbf{B}^{(\mathsf{L})})$ *i.e.* we denote by $\mathbf{B}^{(\mathsf{L})}$ the argmax for $\ell(\cdot)$ for a given support set L. Hence, by the optimality of $\mathbf{B}^{(\mathsf{L}\cup\{i\})}$,

$$\begin{aligned} D(i+1) &= \ell(\mathbf{B}^{(\mathsf{L}\cup\{i\})}) - \ell(\mathbf{B}^{(\mathsf{L})}) \\ &\geq \ell(\mathbf{B}^{(\mathsf{L})} + \alpha\mathbf{u}\mathbf{v}^\top) - \ell(\mathbf{B}^{(\mathsf{L})}) \end{aligned}$$

for an arbitrary $\alpha \in \mathbb{R}$, and the vectors $\mathbf{u}, \mathbf{v}$ selected by `OMPSel`. Using the smoothness of the $\ell(\cdot)$, we get,

$$D(i+1) \geq \alpha\langle \nabla\ell(\mathbf{B}^{(\mathsf{L})}), \mathbf{u}\mathbf{v}^\top\rangle - \alpha^2 \frac{\tilde{M}_1}{2}$$

Putting in $\alpha = \frac{\tau}{\tilde{M}_1}\|\nabla\ell(\mathbf{B}^{(\mathsf{L})})\|_2$, and by τ-optimality of `OMPSel`, we get,

$$D(i+1) \geq \frac{\tau^2}{2\tilde{M}_1}\|\nabla\ell(\mathbf{B}^{(\mathsf{L})})\|_2^2$$

Let S^R be obtained from after sequentially orthogonalizing $\mathsf{S}^\star$ w.r.t. S_i. By definition of the operator norm, we further get,

$$\begin{aligned}
D(i+1) &\geq \frac{\tau^2}{2\tilde{M}_1}\|\nabla\ell(\mathbf{B}^{(\mathsf{L})})\|_2^2 \\
&\geq \frac{\tau^2}{2r\tilde{M}_1}\sum_{i\in\mathsf{S}^R}\langle \mathbf{u}_i\mathbf{v}_i^\top, \nabla\ell(\mathbf{B}^{(\mathsf{L})})\rangle^2 \\
&= \|P_{\mathbf{U}_{\mathsf{S}^R}}\nabla\ell(\mathbf{B}^{(\mathsf{L})})P_{\mathbf{V}_{\mathsf{S}^R}}\|_F^2 \\
&\geq \frac{\tau^2 m_{r+k}}{r\tilde{M}_1}\left(\ell(\mathbf{B}^{\mathsf{L}\cup\mathsf{S}^R}) - \ell(\mathbf{B}^{(\mathsf{L})})\right) \\
&\geq \frac{\tau^2 m_{r+k}}{r\tilde{M}_1}\left(\ell(\mathbf{B}^{\mathsf{S}^\star}) - \ell(\mathbf{B}^{(\mathsf{L})})\right) \\
&= \frac{\tau^2 m_{r+k}}{r\tilde{M}_1}B(i)
\end{aligned}$$

□

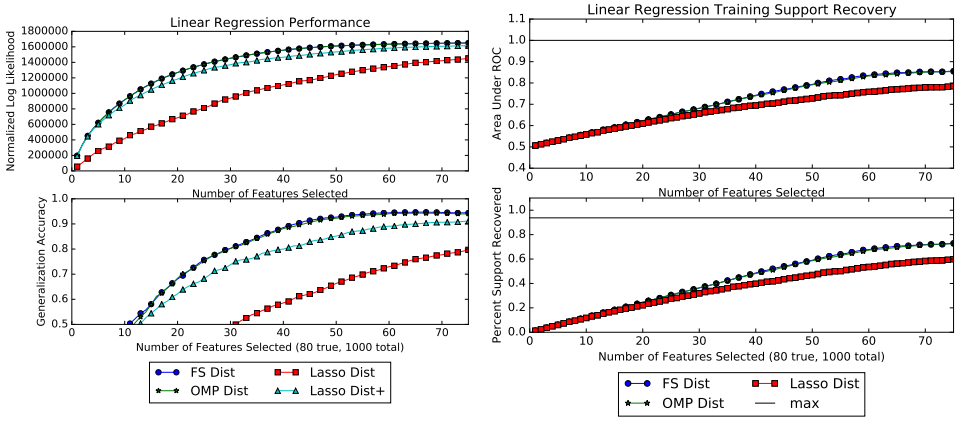
The proof for Theorem 6.3.4 from Lemma 6.3.5 now follows using the same steps as for Theorem 6.3.2 from Lemma 6.3.5.

Remark 6.3.5. Our proof technique for Theorem 6.3.4 can be applied for classical sparsity to improve the bounds obtained by Elenberg et al. [2018] for OMP for support selection under RSC, and by Das and Kempe [2011] for R^2 statistic. If $\tau = 1, r = k$, their bounds involve terms of the form $O(m^2/M^2)$ in the exponent, as opposed to our bounds which only has m/M in the exponent.

Remark 6.3.6. Similar to the greedy algorithm, to achieve a tighter approximation to best rank k solution, one can relax the low rank constraint a little by running the algorithm for $r > k$ greedy iterations. The result obtained by our Theorem 6.3.4 can be compared to the bound obtained by Shalev-Shwartz et al. [2011] [Theorem 2] for the same algorithm. For an ϵ multiplicative approximation, Theorem 6.3.4 implies we need $r/k = O(\log 1/\epsilon)$. On the other hand, Shalev-Shwartz et al. [2011] obtain an additive approximation bound with $r/k = O(1/\varepsilon)$, which is an exponential improvement.

6.4 Experiments

In this section, we empirically evaluate the proposed algorithms.



(a) (b)

Figure 6.1: Distributed linear regression, $l = 10$ partitions, $n = 800$ training and test samples, $\alpha = 0.5$. Results averaged over 10 iterations. Both greedy algorithms outperform ℓ_1 regularization.

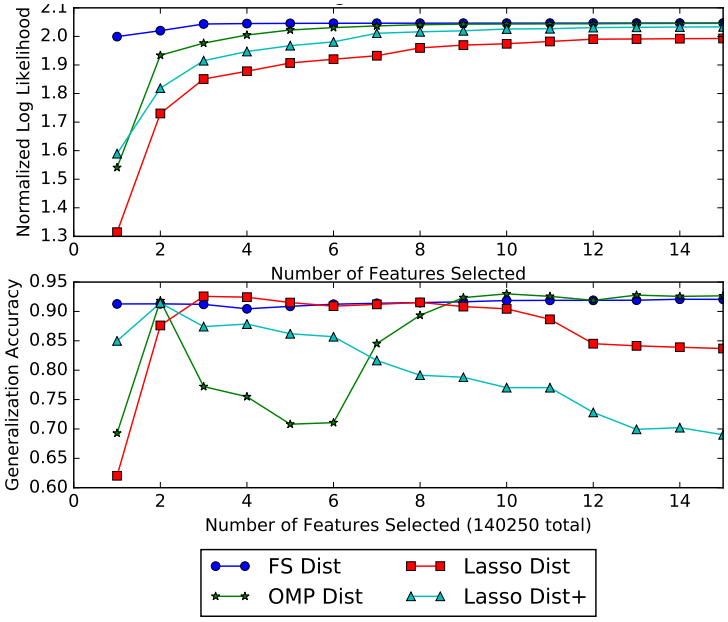


Figure 6.2: Distributed linear regression, Electricity dataset.

6.4.1 Distributed Linear Regression

We consider sparse linear regression in a distributed setting. We generate a 100-sparse regression vector is generated by selecting random nonzero entries of $\boldsymbol{\beta}$,

$$\boldsymbol{\beta}_s = (-1)^{\mathrm{Bern}(1/2)} \times \left(5\sqrt{\frac{\log d}{n}} + \delta_s\right),$$

where δ_s is a standard i.i.d. Gaussian. Measurements $\mathbf{y}$ are taken according to $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{z}$, where $\forall i \in [n], z_i$ is i.i.d. Gaussian with variance set to be $0.01\|\mathbf{X}\boldsymbol{\beta}\|_2^2$. Each row of the design matrix $\mathbf{X}$ is generated by an autoregressive process,

$$X_{n,t+1} = \sqrt{1-\alpha^2}X_{n,t} + \epsilon_{n,t},$$

where $\epsilon_{n,t}$ is i.i.d. Gaussian with variance $\alpha^2 = 0.25$. We take $n = 800$ for the number of both training and test measurements. Results are averaged over 10 iterations, each with a different partition $\{\mathsf{A}_j\}$ of the 1000 features.

We evaluate four variants of DistributedGreedy. The two greedy algorithms are Greedy Forward Selection (FS) and Orthogonal Matching Pursuit (OMP). Lasso sweeps an ℓ_1 regularization parameter λ using LARS [Efron et al., 2004]. This produces nested subsets of features corresponding to a sequence of thresholds for which the support size increases by 1. Lasso uses this threshold, while Lasso+ fits an unregularized linear regression on the support set selected by Lasso.

Figure 6.1 shows the performance of all algorithms on the following metrics: log likelihood (normalized with respect to a null model), generalization to new test measurements from the same true support parameter, area under ROC, and percentage of the true support recovered for $l = 10$.

Next, we run a similar experiment on a large, real-world dataset. We sample $d = 140{,}250$ time series measurements across $n = 370$ customers from the `ElectricityLoadDiagrams` time series dataset [Lichman, 2013]. We consider the supervised learning experiment of predicting the electrical load at the *next* time 140,251. We use half of the customers for training and the rest for testing. Figure 6.2 shows performance of the same algorithms with data distributed across $l = 50$ partitions to select the top $k = 15$ features. We see that distributed Forward Selection produces both largest likelihood and highest generalization score. OMP has second largest likelihood, but its gen-

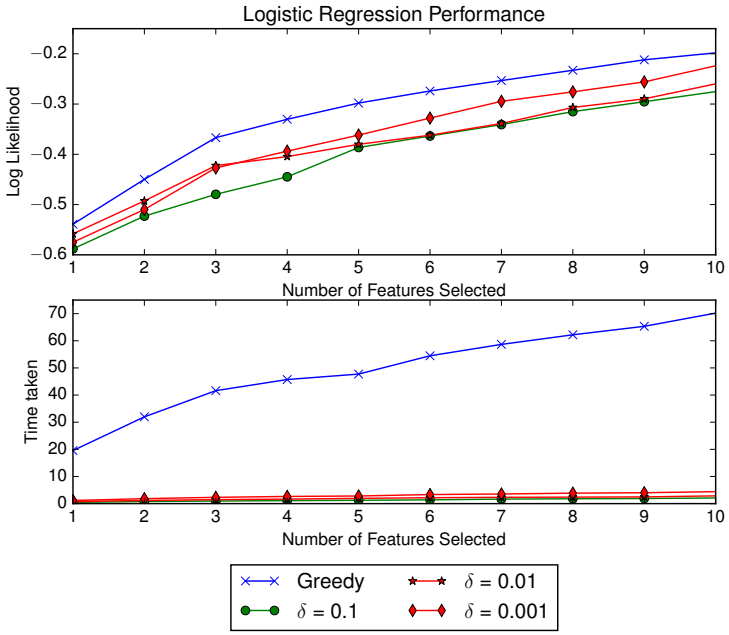


Figure 6.3: Trade off in time vs log likelihood for various values of δ-Stochastic Greedy as opposed to Greedy Forward Selection for logistic regression on `gisette` data [Lichman, 2013]. Results averaged over 10 iterations.

eralization varies widely for different values of k. This is likely due to the random placement of predictive features across a large number of partitions.

6.4.2 Stochastic Sparse Logistic Regression

In this section we demonstrate the applicability of Algorithm 11 for greedy support selection for sparse logistic regression. One would not typically apply the StochasticGreedy algorithm for sparse logistic regression, because the cost function is not submodular. However, the guarantees obtained in Section 6.2 suggest good practical performance which is indeed demonstrated in Figure 6.3. For the experiment we use the `gisette` dataset obtained

from the UCI website [Lichman, 2013]. The dataset is of a handwritten digit recognition problem to separate out digits '4' and '9'. It has 13500 instances, and 5000 features. Figure 6.3 illustrates depicts the tradeoff between the time taken to learn the model and the respective training log likelihood for different values of δ as used in Algorithm 11. As shown, we obtain tremendous speed up with relatively little loss in the log likelihood value even for reasonably large δ values.

6.4.3 Clustering under Stochastic Block Model

First, we test empirically the performance of GECO (Algorithm 13 with $\tau = 1$) for a clustering task. We are provided with a graph with nodes and the respective edges between the nodes. The observed graph is assumed to have been noisily generated from a true underlying clustering. The goal is to recover the underlying clustering structure from the noisy graph provided to us. Our greedy framework is applicable because the adjacency matrix of the true clustering is low rank. We compare performance of Algorithm 13 on simulated data against standard baselines of spectral clustering which are commonly used for this task. We begin by describing a generative model for creating edges between nodes given the ground truth.

The Stochastic Block Model is a model to generate random graphs. It takes its input the set of n nodes, and a partition of $[n]$ which form a set of disjoint clusters, and returns the graph with nodes and the generated edges. The model has two additional parameters, the generative probabilities

(p, q). A pair of nodes within the same cluster have an edge between them with probability p, while a pair of nodes belonging to different clusters have an edge between them with probability q. For simplicity we assume $q = (1 - p)$. The model then iterates over each pair of nodes. For each such pair that belongs to same cluster, it samples an edge as Bernoulli(p), otherwise as Bernoulli($1-p$). This provides us with a $\{0, 1\}$ adjacency matrix.

We compare against two versions of spectral clustering, which is a standard technique applied to find communities in a graph. The method takes as input the $n \times n$ adjacency matrix $\mathbf{A}$, which is a $\{0, 1\}$ matrix with an entry $\mathbf{A}_{ij} = 1$ if there is an edge between node i and j, and is 0 otherwise. From the adjacency matrix, the graph Laplacian $\mathbf{L}$ is constructed. The Laplacian may be unnormalized, in which case it is simply $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix of degrees of nodes. A normalized Laplacian is computed as $\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. After calculating the Laplacian, the algorithm solves for bottom k eigenvectors of the Laplacian, and then apply k-means clustering on the rows of the thus obtained eigenvector matrix. We refer to the works of Shi and Malik [2000], Ng et al. [2001] for the specific details of clustering algorithms using unnormalized and normalized graph Laplacian respectively.

We use our greedy algorithm to cluster the graph by optimizing a logistic PCA objective function, which is a special case of the exponential family PCA [Collins et al., 2001]. For a given matrix $\mathbf{X}$, each entry $\mathbf{X}_{ij}$ is assumed to be independently drawn with likelihood proportional to $\exp \langle \Theta_{ij}, \mathbf{X}_{ij} \rangle - G(\Theta_{ij})$, where Θ is the true underlying parameter, and $G(\cdot)$ is the partition function

corresponding to a generalized linear model (GLM). It is easy to see we can apply our framework of greedy selection by defining $\ell(\cdot)$ as the log-likelihood:

$$\ell(\Theta) = \langle \Theta, \mathbf{X} \rangle - \sum_{i,j} \log G(\Theta_{ij}),$$

where Θ is the true parameter matrix of p and q that generates a realization of $\mathbf{A}$. Since the true Θ is low rank, we get the low rank constrained optimization problem:

$$\max_{\text{rank}(\Theta) \leq k} \ell(\Theta),$$

where k is a hyperparameter suggesting the true number of clusters. Note that lack of knowledge of true value of k is not more restrictive than spectral clustering algorithms which typically also require the true value of k. Having cast the clustering problem in the same form as (6.3), we can apply our greedy selection algorithm as opposed to the more costly alternating minimizing algorithms suggested by Collins et al. [2001].

We generate the data as follows. For $n = 100$ nodes, and fixed number of cluster $k = 5$, we vary the within cluster edge generation probability p from 0.55 to 0.95 in increments of 0.05, and use the Stochastic Block model to generate a noisy graph with each p. Note that smaller p implies that the sampled graph will be more noisy and likely to be more different than the underlying clustering.

We compare against the spectral clustering algorithm using unnormalized Laplacian of Shi and Malik [2000] which we label "Spectral_unnorm$\{k\}$" for $k = \{3, 5, 10\}$, and the spectral clustering algorithm using normalized

Laplacian of Ng et al. [2001] which we label "Spectral_norm$\{k\}$" for $k = \{3, 5, 10\}$. We use Algorithm 13 which we label "Greedy$\{k\}$" for $k = \{3, 5, 10\}$. For each of these models, the referred k is the supplied hyperparameter. We report the least squares error of the output from each model to the true underlying Θ (generalization error), and to the instantiation used for training $\mathbf{X}$ (reconstruction error).

Figure 6.4 shows that the greedy logistic PCA performs well in not only recreating the given noisy matrix (reconstruction) but also captures the true low rank structure better (generalization). Further, note that providing the true hyperparameter k is vital for spectral clustering algorithms, while on the other hand greedy is less sensitive to k. This is very useful in practice as k is typically not known. Spectral clustering algorithms typically select k by computing an SVD and rerunning k-means for different values of k. In addition to being more robust, our greedy algorithm does not need to be rerun for different values of k – it produces solutions incrementally.

6.4.4 Word Embeddings

Algorithms for embedding text into a vector space yield representations that can be quite beneficial in many applications, *e.g.* features for sentiment analysis. Mikolov et al. [2013b] proposed a context-based embedding called skip-gram or `word2vec`. The context of a word can be defined as a set of words before, around, or after the respective word. Their model strives to find an embedding of each word so that the representation predicts the embedding of

each context word around it. Levy and Goldberg [2014] subsequently showed that the word embedding model proposed by Mikolov et al. [2013b] can be reinterpreted as matrix factorization of the *PMI* matrix constructed as follows. A word c is in context of w if it lies within the respective window of w. The PMI matrix is then calculated as

$$\mathrm{PMI}_{w,c} = \log\left(\frac{p(w,c)}{p(w)p(c)}\right).$$

In practice the probabilities $p(w,c), p(w), p(c)$ are replaced by their empirical counterparts. Further, note that $p(w,c)$ is 0 if words c and w do not coexist in the same context, which yields $-\infty$ for PMI. Levy and Goldberg [2014] suggest using an alternative: $\mathrm{PPMI}_{w,c} = \max\{\mathrm{PMI}_{w,c}, 0\}$. They also suggest variations of PMI hyper parameterized by k which corresponds to the number of negative samples in the training of the original skip gram model.

We employ the binomial PCA model on the normalized count matrix (instead of the PMI), in a manner similar to the clustering approach in Section 6.4.3. The normalized count matrix is calculated simply as $\frac{p(w,c)}{p(w)}$, without taking logarithms. This gives us a probability matrix which has each entry between 0 and 1, and which can be factorized under the binomial model greedily as per Algorithm 13.

We empirically study the embeddings obtained by binomial factorization on two tasks – word similarity and analogies. For word similarity, we use the W353 dataset [Finkelstein et al., 2001] and the MEN data [Bruni et al., 2012]. Both these datasets contain words with human assigned sim-

ilarity scores. We evaluate the embeddings by their cosine similarity, and measuring the correlation with the available human ratings. The fraction of correctly answered queries are returned as the metric. For the analogy task, we use the Microsoft Research (MSR) syntactic analogies [Mikolov et al., 2013c] and the Google mixed analogies dataset [Mikolov et al., 2013a]. For completing analogy a:b::c:x, the prediction is calculated as $\arg\max_x \frac{\cos(c,x)\cos(b,x)}{\cos(a,x)}$. To compute accuracy, we use the multiplication similarity metric as used by Levy and Goldberg [2014]. To train the word embeddings, we use the 2013 news crawl dataset[1]. We filter out stop words, non-ASCII characters, and words occurring less than 2000 times (which yields a vocabulary of 6713). Note that since we keep only the most common words, several queries from the datasets are invalid because we do not have embeddings for words appearing in them. However, we do include them by assigning invalid queries a value of 0 and reporting the overall average over the entire dataset.

Table 6.1 shows the empirical evaluation. SVD and PPMI are the models proposed by Levy and Goldberg [2014], while SGNS is the skipgram with negative sampling model of Mikolov et al. [2013b]. We run each of these for $k = \{5, 10, 15, 20\}$ and report the best results. This shows that alternative factorizations such as our application of binomial PCA can be more consistent and competitive with other embedding methods.

[1] `http://www.statmt.org/wmt14/training-monolingual-news-crawl`

Table 6.1: Empirical study of binomial based greedy factorization shows competitive performance of word embeddings of common words across tasks and datasets.

	W353	MEN	MSR	Google
# Queries	353	3000	8000	19544
SVD	0.226	0.233	0.086	0.092
PPMI	0.175	0.178	0.210	0.130
SGNS	0.223	0.020	0.052	0.002
Greedy	0.202	0.198	0.176	0.102

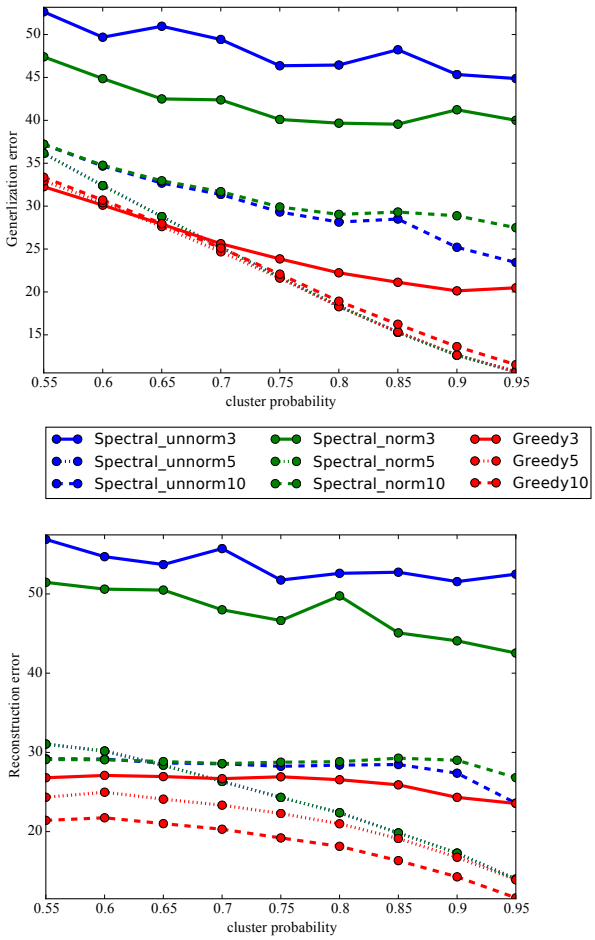


Figure 6.4: Greedy Logistic PCA vs spectral clustering baselines averaged over 10 runs. *Top*: Robust performance of greedy logistic PCA for generalizing over varying values of k across different values of p, spectral clustering algorithms are more sensitive to knowing true value of k *Bottom*: Strong performance of greedy logisitic PCA even with small value of $k = 3$ for reconstructing the given cluster matrix.

Chapter 7

Conclusion and Future work

The focus of this dissertation was to develop novel frameworks to use Bayesian approximation employing greedy algorithm variants to obtain parsimonious distributions. The underlying theme was not to just provide scalable algorithms that are efficient and work well in practice, but also provide worst case theoretical guarantees that motivate application to problems beyond those discussed in this dissertation. Indeed, most of the presented theoretical studies are general enough to be applicable to settings beyond those presented in this work. These include submodularity of the KL projection (Section 3.3.1), submodularity of MMD selection (Section **??**), weak submodularity of Sequential Bayesian Quadrature (Section 6.3.2), and all the results on large scale algorithms and low rank approximation presented in Chapter 6. These theoretical studies open avenues to several potential areas of future research for other cost functions for generalized VI. For the case of the weakly submodular functions, it was shown that submodularity is too strong a requirement to obtain reasonable approximation guarantees. This obviously begets the question – what other properties of submodular functions are redundant ?

On the practical side, extensive empirical evaluation was presented to il-

lustrate the effectiveness of the proposed methods in quantitative metrics when compared to respective established baselines for sparse learning. The tested models include group sparse bayesian Regression, structured sparse probabilistic PCA, sparse CMF, sparse logistic regression etc. The applicability of the developed methods extend far beyond than those that were tested because of the provided approximation guarantees. As such, guidelines were provided that allow easy extension and application to other potential applications such as tree sparse regression, or Canonical Correlation Analysis (CCA).

Similarly, with the recent interest in the machine learning community on model interpretation, it is evident that this dissertation has barely scratched the surface. Interesting applications of Fisher kernels for screening adversarial examples, outlier detection and possibly design of robust machine learning models are natural next steps.

Bibliography

A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications, IOS Press*, pages 39–59, 1994.

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. *CoRR*, 2016.

Yasemin Altun and Alexander J. Smola. Unifying divergence minimization and statistical inference via convex duality. In *Conference on Learning Theory (COLT)*, pages 139–153, 2006.

Cédric Archambeau and Francis R. Bach. Sparse probabilistic projections. In *Neural Information Processing Systems (NIPS)*, pages 73–80, 2008.

Francis R. Bach and Michael I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical report, UC Berkeley, 2005.

Rafael Barbosa, Alina Ene, Huy Le Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets.

In *International Conference on Machine Learning (ICML)*, pages Pages 1236–1244, 2015.

Isabelle Bichindaritz and Cindy Marling. Case-based reasoning in the health sciences: What's next? *Artificial Intelligence in Medicine*, 36(2):127–135, February 2006.

J. Bien and R. Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, pages 2403–2424, 2011.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 136–145, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Gruia Calinescu, Chandra Chekuri, Martin Pl, and Jan Vondrk. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Knowlege Discovery and Data Mining (KDD)*, pages 1721–1730, 2015.

Carlos M Carvalho, Nicholas G Polson, and James G Scott. The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480, 2010.

Joseph T Chang and David Pollard. Conditioning as disintegration. *Statistica Neerlandica*, 51(3):287–317, 1997.

Yutian Chen, Max Welling, and Alexander J. Smola. Super-samples from kernel herding. In *Uncertainty in Aritifical Intelligence (UAI)*, pages 109–116, 2010.

M.S. Cohen, J.T. Freeman, and S. Wolf. Metarecognition in time-stressed decision making: Recognizing, critiquing, and correcting. *Human Factors*, 38(2):206–219, 1996.

Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. A generalization of principal component analysis to the exponential family. In *Neural Information Processing Systems (NIPS)*, pages 617–624. MIT Press, 2001.

Paul Damien and Stephen G Walker. Sampling truncated normal, beta, and gamma densities. *Journal of Computational and Graphical Statistics*, 10(2): pages 206–215, 2001.

Abhimanyu Das and David Kempe. Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection. In *International Conference on Machine Learning (ICML)*, pages 1057–1064, 2011.

Alexandre d'Aspremont, Francis R. Bach, and Laurent El Ghaoui. Full regularization path for sparse principal component analysis. In *International Conference on Machine Learning (ICML)*, pages 177–184, 2007.

J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

Josef Dick and Friedrich Pillichshammer. *Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration*. Cambridge University Press, New York, NY, USA, 2010.

Miro Dudik, Zaid Harchaoui, and Jerome Malick. Lifted coordinate descent for learning with trace-norm regularization. In *Artifical Intelligence and Statistics (AISTATS)*, pages 327–336, 2012.

Bradley Efron, Trevor Hastie, Ian Johnstone, and Robert Tibshirani. Least Angle Regression. *Annals of Statistics*, 32(2):407–499, 2004.

Ethan R. Elenberg, Rajiv Khanna, Alexandros G. Dimakis, and Sahand Negahban. Restricted Strong Convexity Implies Weak Submodularity. *Annals of Statistics*, 2018.

Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: the concept revisited. pages 406–414, 2001.

A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian data analysis*. Taylor & Francis, 2014.

Krzysztof J. Gorgolewski, Gael Varoquaux, Gabriel Rivera, Yannick Schwarz, Satrajit S. Ghosh, Camille Maumet, Vanessa V. Sochat, Thomas E. Nichols, Russell A. Poldrack, Jean-Baptiste Poline, Tal Yarkoni, and Daniel S. Margulies. Neurovault.org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain. *Frontiers in Neuroinformatics*, 9:8, 2015.

A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, page 723773, 2012.

Rémi Gribonval and P Vandergheynst. On the exponential convergence of matching pursuits in quasi-incoherent dictionaries. *IEEE Trans. Inform. Theory*, 52(1):255–261, 2006.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

Jonathan H. Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. In *Neural Information Processing Systems (NIPS)*, pages 4087–4095, 2016.

J.J. Hull. A database for handwritten text recognition research. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(5):Page 550–554, 1994.

Ferenc Huszar and David K. Duvenaud. Optimally-weighted herding is bayesian quadrature. In *Uncertainty in Artificial Intelligence (UAI)*, pages 377–386, 2012.

Hemant Ishwaran and J Sunil Rao. Spike and slab variable selection: frequentist and bayesian strategies. *Annals of Statistics*, pages 730–773, 2005.

Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Neural Information Processing Systems (NIPS)*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

Martin Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In *International Conference on Machine Learning (ICML)*, pages 427–435, 2013.

Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *International Conference on Machine Learning (ICML)*, pages 471–478. Omnipress, 2010.

E. T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review Online Archive (Prola)*, 106(4):620–630, 1957.

R. Jenatton, G. Obozinski, and F. Bach. Structured sparse principal component analysis. In *Artificial Intelligence and Statistics (AISTATS)*, pages 396–407, 2010.

Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Jour-*

nal of Machine Learning Research (JMLR), 11:517–553, March 2010. ISSN 1532-4435.

Rajiv Khanna, Ethan R. Elenberg, Alexandros G. Dimakis, Sahand Neghaban, and Joydeep Ghosh. Scalable Greedy Support Selection via Weak Submodularity. *Artificial Intelligence and Statistics (AISTATS)*, pages 1560–1568, 2017.

B. Kim, C. Rudin, and J.A. Shah. The Bayesian Case Model: A generative approach for case-based reasoning and prototype classification. In *Neural Information Processing Systems (NIPS)*, pages 1952–1960, 2014.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Neural Information Processing Systems (NIPS)*, pages 2280–2288. 2016.

Arto Klami, Guillaume Bouchard, and Abhishek Tripathi. Group-sparse embeddings in collective matrix factorization. *CoRR*, abs/1312.5921, 2013a.

Arto Klami, Seppo Virtanen, and Samuel Kaski. Bayesian canonical correlation analysis. *Journal of Machine Learning Research (JMLR)*, 14(1): 965–1003, April 2013b. ISSN 1532-4435.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1885–1894, 2017.

A. N. Kolmogorov. *Foundations of the theory of probability.* Chelsea, New York, 1933.

Oluwasanmi Koyejo. Constrained relative entropy minimization with applications to multitask learning. *PhD Thesis, University of Texas at Austin, Austin, Texas*, 2013.

Oluwasanmi Koyejo and Joydeep Ghosh. Constrained bayesian inference for low rank multitask learning. *arXiv preprint arXiv:1309.6840*, 2013.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284, June 2008.

Solomon Kullback. Information theory and statistics. Dover, 1959.

L. I. Kuncheva and J.C. Bezdek. Nearest prototype classification: clustering, genetic algorithms, or random search? *IEEE Transactions on Systems, Man, and Cybernetics*, 28(1):160–164, 1998.

Simon Lacoste-Julien and Martin Jaggi. On the Global Linear Convergence of Frank-Wolfe Optimization Variants. In *NIPS 2015*, pages 496–504, 2015.

Yann LeCun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Neural Information Processing Systems (NIPS)*, pages 2177–2185. 2014.

Moshe Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Jun Liu, Shuiwang Ji, and Jieping Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

J. R. Lloyd and Z. Ghahramani. Statistical model criticism using kernel two sample tests. In *Neural Information Processing Systems (NIPS)*, pages 829–837, 2015.

Francesco Locatello, Rajiv Khanna, Michael Tschannen, and Martin Jaggi. A unified optimization view on generalized matching pursuit and frank-wolfe. In *Artificial Intelligence and Statistics (AISTATS)*, pages 860–868, 2017.

Po-Ling Loh and Martin J. Wainwright. Regularized m-estimators with nonconvexity: Statistical and algorithmic theory for local optima. *Journal of Machine Learning Research (JMLR)*, 16(1):559–616, January 2015.

David MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

Mokshay Madiman and Prasad Tetali. Information inequalities for joint distributions, with interpretations and applications. *IEEE Transactions on Information Theory*, 56(6):2699–2713, 2010.

Julien Mairal, Francis R. Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *International Conference on Machine Learning (ICML)*, pages 689–696, 2009.

Vincent Michel, Alexandre Gramfort, Gaël Varoquaux, Evelyn Eger, Christine Keribin, and Bertrand Thirion. A supervised clustering approach for fMRI-based inference of brain states. *Pattern Recognition*, 45(6):2041–2049, 2012.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013b.

Tomas Mikolov, Scott Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751. Association for Computational Linguistics, May 2013c.

Vahab Mirrokni and Morteza Zadimoghaddam. Randomized Composable Core-sets for Distributed Submodular Maximization. In *STOC '15*, pages 153–162, New York, New York, USA, 2015. ACM Press.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause 0001. Distributed Submodular Maximization - Identifying Representative Ele-

ments in Massive Data. *Neural Information Processing Systems (NIPS)*, pages 2049–2057, 2013.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1812–1818, 2015.

Toby J Mitchell and John J Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

Radford Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

Sahand Negahban, Pradeep Ravikumar, Bin Yu, and Martin J. Wainwright. A Unified Framework for High-Dimensional Analysis of M-Estimators with Decomposable Regularizers. *Statistica Sinica*, 27(4):538–557, 2012.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

A. Newell and H.A. Simon. *Human problem solving*. Prentice-Hall Englewood Cliffs, 1972.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems (NIPS)*, pages 849–856. MIT Press, 2001.

A O'Hagan. Bayes-Hermite Quadrature. volume 29. Journal of Statistical Planning and Inference, Nov 1991.

Dimitris S. Papailiopoulos, Alexandros G. Dimakis, and Stavros Korokythakis. Sparse PCA through low-rank approximations. *International Conference on Machine Learning (ICML)*, pages 747–755, 2013.

Mijung Park and Jonathan W Pillow. Bayesian inference for low rank spatiotemporal neural receptive fields. In *Neural Information Processing Systems (NIPS)*, pages 2688–2696. 2013.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

Russell A. Poldrack, Jeanette Mumford, and Thomas Nichols. *Handbook of Functional MRI Data Analysis*. Cambridge University Press, Jan 2011. ISBN 9780521517669.

Russell .A. Poldrack, T Laumann, O Koyejo, B Gregory, A Hover, M Chen, J Luci, S Joo, D Handwerker, J Liang, R Boyd, S Hunicke-Smith, Z Simpson,

T Caven, V Sochat, J Shine, E Gordon, A Snyder, B Adeyemo, S Petersen, D Glahn, D McKay, J Curran, H Gring, M Carless, J Blangero, L Frick, E Marcotte, and J Mumford. Long-term neural and physiological phenotyping of a single human. *Nature Communications*, 9(8885), 2015.

Christian P Robert, George Casella, and Chritian P Robert. *Monte Carlo statistical methods*, volume 58. Springer New York, 1999.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

S Shalev-Shwartz, A Gonen, and O Shamir. Large-scale convex minimization with a low-rank constraint. In *International Conference on Machine Learning (ICML)*, pages 329–336, 2011.

D. Sharma, A. Kapoor, and A. Deshpande. On greedy maximization of entropy. In *International Conference on Machine Learning (ICML)*, pages 1330–1338, 2015.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation.

IEEE Trans. Pattern Anal. Mach. Intell., 22(8):888–905, August 2000. ISSN 0162-8828.

Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

Ajit Paul Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *Knowledge Discovery and Data Mining (KDD)*, pages 650–658. ACM, 2008.

Stephen M Smith, Peter T Fox, Karla L Miller, David C Glahn, P Mickle Fox, Clare E Mackay, Nicola Filippini, Kate E Watkins, Roberto Toro, Angela R Laird, et al. Correspondence of the brain's functional architecture during activation and rest. *Proceedings of the National Academy of Sciences*, 106 (31):13040–13045, 2009.

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Jussi Tohka, Karin Foerde, Adam R Aron, Sabrina M Tom, Arthur W Toga, and Russell A Poldrack. Automatic independent component labeling for artifact removal in fmri. *Neuroimage*, 39(3):1227–1245, 2008.

Dimitris G Tzikas, CL Likas, and Nikolaos P Galatsanos. The variational approximation for bayesian inference. *Signal Processing Magazine, IEEE*, 25(6):131–146, 2008.

David C. Van Essen, Stephen M. Smith, Deanna M. Barch, Timothy E.J. Behrens, Essa Yacoub, and Kamil Ugurbul. The wu-minn human connectome project: An overview. *NeuroImage*, 80:62 – 79, 2013. ISSN 1053-8119. Mapping the Connectome.

K.R. Varshney. Engineering safety in machine learning. *CORR/arXiv:1601.04126*, 2016.

Peter M Williams. Bayesian conditionalisation and the principle of minimum information. *The British Journal for the Philosophy of Science*, 31(2):131–144, 1980.

David P Wipf and Srikantan S Nagarajan. A new view of automatic relevance determination. In *Neural Information Processing Systems (NIPS)*, pages 1625–1632, 2007.

Daniela M. Witten, Trevor Hastie, and Robert Tibshirani. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515 – 534, 2009.

Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research (JMLR)*, 14(1): 899–925, April 2013.

Vita

Rajiv Ashu Khanna completed his undergraduate studies in Computer Science and Engineering at National Institute of Technology at Jallandhar in India in June 2006. He received his Masters in Computer Science and Engineering at Indian Institute of Technology at Mumbai in India in July 2008. During his Masters, his thesis was on developing novel algorithms for structural SVMs. He worked at Yahoo! Labs Bangalore from 2008 to 2012 as a Research Engineer. His job was at an intersection of science and engineering, with responsibilities including developing scalable prototypes for new technologies in collaboration with scientists and pushing vetted tech to production in collaboration with engineers. In Fall 2012, he joined University of Texas at Austin as a graduate student. His research interests include approximate inference, scalable algorithms for discrete optimization and interpretability in machine learning.

Permanent address: The University of Texas at Austin, Austin, TX
78712

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.